

PRACTICAL SECURE COMPUTATION FOR INTERNET INFRASTRUCTURE

at the
Computer Science Department
of Technische Universität Darmstadt

Dissertation
of
Kris Shrishak

Submitted in fulfilment of the requirements for the
degree of Doktoringenieur
(Dr.-Ing.)



Referees: Prof. Dr. Michael Waidner
Prof. Dr. Jean-Pierre Seifert

Date of submission: 17.12.2020
Date of oral exam: 22.04.2021

Darmstadt, 2021
D 17

This document is provided by tuprints, e-publishing service of TU Darmstadt.

<http://tuprints.ulb.tu-darmstadt.de>

tuprints@ulb.tu-darmstadt.de

Please cite the document as:

URN: [urn:nbn:de:tuda-tuprints-185043](https://nbn-resolving.org/urn:nbn:de:tuda-tuprints-185043)

URL: <https://tuprints.ulb.tu-darmstadt.de/id/eprint/18504>

Kris Shrishak. *Practical Secure Computation for Internet Infrastructure*, Dissertation, Technische Universität Darmstadt, 2021.

This work is licensed under CC BY-NC-ND 4.0. To view a copy of this license, visit

<https://creativecommons.org/licenses/by-nc-nd/4.0>



Abstract

The Internet has been a boon in the lives of many in the world, opening up opportunities that may have been unknown or inaccessible to them. The growth in the availability of computational resources has made it possible to collect, compile, store, process and interpret data at a scale that was not imaginable in the past. The combination of the Internet and computing resources has resulted in a world that creates more data every year than ever in the past, where data can be harvested for the benefit of society. However, when the surface seems too shiny, the dangers lurk nearby. One such danger is privacy violation that can take several forms including nosy corporate employees, hacked databases as well as government coercion of centralised authorities that manage the Internet infrastructure.

Secure multi-party computation (MPC) is a cryptographic tool for privacy-preserving computation. MPC allows multiple entities to perform joint computation over their private inputs, revealing only the output. Although the theoretical foundations for the two-party variant, secure two-party computation (2PC), were introduced in the 1980s, MPC has not yet seen widespread deployment in spite of its benefits. Not only is MPC useful when data needs to be processed, but it is also useful when cryptographic data such as signing keys are to be kept securely.

In this thesis, we make MPC practical to secure Internet infrastructure. While MPC has been applied to many applications, it has not yet been used to secure Internet infrastructure. In the process of making MPC practical, we address several challenges in this thesis. First, we observe that the practical performance of 2PC can be improved by the use of different transport layer protocols. On the basis of this observation, we develop a framework that automates the integration of transport layer protocols into 2PC implementations. We show through extensive evaluations that the efficiency gained by using better transport layer protocols is sometimes much greater than that can be achieved by using stronger security assumptions.

Second, we observe a practical security issue where mechanisms to secure fundamental protocols of the Internet infrastructure, such as routing and domain name system, rely on centralised authorities. In particular, signing keys that should be held by domain owners and Internet number resource owners in security mechanisms for Internet infrastructure are instead outsourced to centralised authorities. Nevertheless, vulnerabilities as well as conflict of interests often make the requirement for trust unsuitable for practical purposes. We replace trust in centralised authorities by designing systems that use MPC and distribute trust.

Finally, we design and implement efficient threshold signature protocols, a specific instance of MPC, that we use to improve the security of Internet infrastructure. Our design uses a generic transformation to turn essentially any MPC protocol into an equally secure and efficient protocol that computes signatures in a threshold setting. Our design is the first to support preprocessing (independent of the message being signed as well as the key being used to sign), which is crucial for practical efficiency as it adds minimal overhead compared to the approach of centralised authorities being in charge of the keys.

List of Publications

Papers in conferences and workshops with proceedings that are part of this thesis are mentioned here. Author names are listed alphabetically in these publications.

- [1] Kris Shrishak and Haya Shulman. Privacy preserving and resilient RPKI. In *40th IEEE Conference on Computer Communications, INFOCOM 2021, Virtual Conference, May 10-13, 2021*. IEEE, 2021.
- [2] Anders P. K. Dalskov, Claudio Orlandi, Marcel Keller, Kris Shrishak, and Haya Shulman. Securing DNSSEC keys via threshold ECDSA from generic MPC. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, Guildford, UK, September 14-18, 2020, Proceedings, Part II*, volume 12309 of *Lecture Notes in Computer Science*, pages 654–673. Springer, 2020.
- [3] Kris Shrishak and Haya Shulman. Limiting the power of RPKI authorities. In *ANRW '20: Applied Networking Research Workshop, Virtual Event, Spain, July 27-30, 2020*, pages 12–18. ACM, 2020.
- [4] Markus Brandt, Claudio Orlandi, Kris Shrishak, and Haya Shulman. Optimal transport layer for secure computation. In Pierangela Samarati, Sabrina De Capitani di Vimercati, Mohammad S. Obaidat, and Jalel Ben-Othman, editors, *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, ICETE 2020 - Volume 2: SECRYPT, Lieusaint, Paris, France, July 8-10, 2020*, pages 130–141. ScitePress, 2020.
- [5] Kris Shrishak and Haya Shulman. MPC for securing internet infrastructure. In *50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020 - Supplemental Volume*, pages 47–48. IEEE, 2020.
- [6] Kris Shrishak, Haya Shulman, and Michael Waidner. Removing the bottleneck for practical 2PC. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 2300–2302. ACM, 2018.

Contributions

This thesis is a culmination of years of work, work that has extensively benefited from discussion and collaboration with some of the most thoughtful researchers I have come across. The diversity of knowledge that my collaborators brought to the table accelerated the work and contributed to the publications that form the core of this thesis. In some cases, our contributions are closely integrated and it is hard to identify individual contributions. Nevertheless, the contributions in joint papers is stated along with the outline of this thesis. That said, this is the only section of the thesis where first-person singular personal pronouns are used.

Chapter 1 introduces the contributions of this thesis and is partly adapted from a publication with Haya Shulman [5].

Chapter 4 is based on two publications and includes the contribution of this thesis related to practical efficiency of 2PC. The first publication [6] is a joint work with Haya Shulman and Michael Waidner and the second publication [4] is a joint work with Markus Brandt, Claudio Orlandi and Haya Shulman. In the latter publication, I contributed to the analysis of the 2PC protocols and made a major contribution to the design of the framework. The analysis of the transport layer protocols was jointly performed with Markus. Furthermore, I evaluated the performance of the protocols in our framework. Markus implemented the transport wrapper that we use in the framework.

Chapter 5 includes two contributions: development of efficient threshold signatures and securing domain name system security extensions (DNSSEC) keys. It is based on a joint work with Anders Dalskov, Marcel Keller, Claudio Orlandi and Haya Shulman [2]. My contribution took the form of identifying the problem with DNSSEC deployments before designing the multiparty zone signing system. I performed the measurements to quantify the extent to which domains use multiple operators on the Internet. Anders and I contributed to the construction of our

generic threshold Elliptic Curve Digital Signature Algorithm protocol. Anders also performed the evaluations.

Chapter 6 includes the contribution of this thesis related to distributed resource public key infrastructure (RPKI). It is based on two works with Haya Shulman [1, 3]. My contribution took the form of identifying the problem with RPKI deployments before designing the distributed RPKI system with a stronger threat model and proposed two deployment models. I used RPKI data to understand the efficiency requirements of the system and performed extensive evaluations of the system to show that our system satisfies the requirements.

Acknowledgements

*It's like driving a car at night.
You never see further than your
headlights, but you can make the
whole trip that way.*

E.L. Doctorow

I am indebted to the many scholars whose ideas this thesis builds on. I owe a debt of gratitude to the people who have supported me through the years that I have worked on the research that has culminated in this thesis. While I mention some of them here, I hope the rest know that I appreciate them.

I thank my supervisors Haya Shulman and Michael Waidner for guiding and encouraging me while giving me the freedom to explore ideas. Special thanks to Jean-Pierre Seifert for being my second referee. I have had the opportunity to learn a lot from Claudio Orlandi. Thank you for hosting me in Aarhus for a research visit and for the opportunity to meet some wonderful people. Thanks to Marc Fischlin, Max Mühlhäuser, and Felix Wolf for taking the time to serve on the committee.

This thesis would not have taken its present shape but for the assistance of many people. I thank my collaborators—Markus Brandt, Anders Dalskov, Marcel Keller, Claudio Orlandi, Haya Shulman, and Michael Waidner; umpteen number of people from reviewers and sub-reviewers to program committee members of conferences who keep the academic publishing system working; Birgit Blume, Karina Köhres, and Christine Roth for taking care of many of the administrative tasks; Sabrina Glindmeyer at TU Darmstadt and the employees of Ausländerbehörde Darmstadt who have made my interaction with the state bureaucracy a pleasant experience.

Research requires a lot of resources and tools. I thank the *Deutsche Forschungsgemeinschaft* for funding me through GRK Privacy and Trust for Mobile Users.

This thesis has benefited from a number of open source projects: Git, GnuPG, GnuTLS, IACR ePrint Archive, Knot DNS, L^AT_EX, Linux, Mozilla Firefox, Mozilla Thunderbird, Numpy, OpenSSL, OpenSSH, Python, Ubuntu, Wikimedia, Wireshark, and many more. I thank the people who develop and maintain open source projects.

When one moves to a new city, one is faced with a conundrum of needing help to find one's feet and not knowing whom to ask. I was fortunate to have the assistance of Laura Keil, Daniel Senf, and Hervais-Clemence Simo Fhom who went out of their way to help me settle into a new city. They have remained patient with me through our many walks and conversations, and have turned into wonderful friends of mine. Darmstadt has also provided me with an environment to meet and engage in memorable conversations with Jacqueline Brendel, Max Maass, Sogol Mazaheri, Olga Sanina, and Markus Uhlmann. ESOC theatre group has welcomed me and given me the opportunity to learn so much about theatre. Special thanks to Sarah Blake, Gerrit-Milena Falker, and Luke Lucas.

Books have been an important part of my life. The library of TU Darmstadt, *Universitäts- und Landesbibliothek*, has been one of my favourite places in Darmstadt. Their collection that has covered a wide range of my interests has sustained me during my time in Darmstadt. During my travels, Harper's Bookstore Tel Aviv, Sellers & Newel in Toronto, Tulibris in Brussels, and many more bookshops have welcomed me. Some of the shops were hard to find and even harder to leave.

Finally, I thank my friends scattered around the world who have shared time with me. While I have mentioned some of them already, I owe a special mention to three incredible individuals who are probably the best listeners I have come across and who have supported me during uncertain times: Shruti Devasenapathy, Leroy Oostenbrink, and Klaas Pieter van der Tempel.

Kris Shrishak

Darmstadt, December 2020

Contents

Abstract	iii
Contents	xi
List of Figures	xv
List of Tables	xvii
Acronyms and Initialisms	xix
1 Introduction	1
1.1 Secure Multi-party Computation	2
1.2 Contributions	3
1.3 Thesis Outline	6
2 Secure Computation	7
2.1 Adversarial Models	7
2.2 Oblivious Transfer	8
2.3 Secure Computation Constructions	9
2.3.1 2PC Based on Garbled Circuits	9
2.3.2 Secret Sharing	11
2.4 Threshold Signatures	13
3 Network Protocols	15
3.1 Transport Layer Protocols	15
3.1.1 TCP-CUBIC	16
3.1.2 SABUL	16
3.2 Domain Name System	17
3.2.1 Domain Name System Security Extensions	17
3.2.2 Organizations in DNS signing setup	18
3.2.3 Zone Signing	18
3.2.4 Registrar as DNS Operator	20

3.2.5	Multi-Operator Setting	20
3.3	Routing	20
3.3.1	Resource Public Key Infrastructure	21
3.3.2	Hosted and Delegated RPKI	22
3.3.3	Route Origin Validation	23
4	Optimal Transport Layer for Secure Computation	25
4.1	Transputation Framework	29
4.1.1	Components	29
4.1.2	Usage of Transputation	31
4.2	Secure Computation Layer	32
4.2.1	Yao vs. GMW	32
4.2.2	Garbling Schemes	33
4.2.3	Applications and Circuit Size	36
4.3	Transport Layer	37
4.3.1	Transport Protocol Selection	38
4.3.2	Transport Protocols in Transputation	39
4.4	Transputation Implementation	40
4.5	Simulations and Evaluations	42
4.5.1	Simulations	42
4.5.2	Deployment Setups	47
4.5.3	Experimental Evaluations	48
4.6	Related Works	50
5	Securing DNSSEC Keys via Threshold ECDSA From Generic MPC	53
5.1	Quantifying Multiple Operators	57
5.1.1	Data Collection Methodology	57
5.1.2	Data Analysis	59
5.2	System and Threat Model	60
5.2.1	System and Communication Model	60
5.2.2	Threat Model	61
5.3	Threshold ECDSA	61
5.3.1	ECDSA	61
5.3.2	Secure Computation on Groups	62
5.3.3	Active Security using SPDZ like MACs	65
5.3.4	Multiparty ECDSA Protocol using ABB+	67
5.3.5	Security Analysis	68
5.4	Multiparty Zone Signing System	68
5.4.1	Setup	70
5.4.2	Key Generation/Rollover	70
5.4.3	Signing	71

5.5	Evaluation	73
5.5.1	Protocols	74
5.5.2	MASCOT- Optimizations	74
5.5.3	Comparison with Prior Work	75
5.5.4	Key Generation	76
5.5.5	Amortizing Signing	76
5.5.6	Overhead for Operators	76
5.6	Related Works	77
6	Distributed RPKI	81
6.1	System and Threat Model	84
6.1.1	Threat Model	84
6.1.2	System and Communication Model	86
6.2	Distributed RPKI	86
6.2.1	System Setup	86
6.2.2	DRPKI Protocol Phases	88
6.2.3	Deployment Scenarios	92
6.3	Implementation and Evaluation	94
6.3.1	Deployment Setups	95
6.3.2	Experimental Evaluations	96
6.4	Analysis	97
6.5	Related Works	99
7	Summary and Future Work	101
	Bibliography	105

List of Figures

1.1	Secure multiparty computation.	3
3.1	Route origin authorization.	22
4.1	TCP vs. SABUL as garbled circuit size increases.	29
4.2	Transputation framework.	30
4.3	Setting up a server with our wrapper.	41
4.4	Example of an echo client using our wrapper.	42
4.5	Effect of latency on 2PC applications.	44
4.6	Effect of loss on 2PC applications.	45
4.7	Performance of JustGarble protocol at various bandwidths.	46
4.8	Performance of GLNP15 protocol at various bandwidths.	46
4.9	Performance of SHA256-GLNP15 in different bandwidths and latencies.	46
4.10	Comparison of assumptions.	49
4.11	Positioning our work within related work.	51
5.1	Proportion of domains with multiple operators.	59
5.2	Arithmetic black-box functionality.	63
5.3	Extended arithmetic black-box.	65
5.4	Full protocol for threshold ECDSA signatures using ABB+.	69
5.5	Setup at DNS operator.	70
5.6	Zone signing.	72
6.1	System setup.	87
6.2	Distributed RPKI architecture.	87
6.3	Key generation protocol.	88
6.4	Signing protocol.	90
6.5	Latency and bandwidth between servers.	95
6.6	Average number of ROAs added and removed per day.	99
6.7	Maximum number of ROAs added and removed per month.	99

List of Tables

4.1	Yao vs. GMW.	33
4.2	List of implementations.	34
4.3	Number of boolean gates per function.	37
4.4	Circuit size per function for each garbling scheme.	37
4.5	Experimental results for garbled circuits protocols.	49
5.1	RTT between the servers used.	75
5.2	Comparison of our threshold ECDSA protocols with prior work.	77
5.3	Key generation benchmark.	77
5.4	Preprocessing throughput and signing time.	78
6.1	Four MPC protocols for distributed RPKI.	94
6.2	Location of RIRs and AWS virtual machines.	96
6.3	Key generation benchmark.	96
6.4	Breakdown of preprocessing and online signing time.	97
6.5	Throughput for preprocessing and online phases.	98
6.6	Communication per party.	98

Acronyms and Initialisms

2PC	Secure Two-Party Computation
ABB	Arithmetic Black-Box
AES	Advanced E ncryption S tandard
AES-NI	Advanced E ncryption S tandard N ew I nstructions
AFRINIC	AFR ican N etwork I nformation C entre
APNIC	A sia- P acific N etwork I nformation C entre
ARIN	A merican R egistry for I nternet N umbers
AS	A utonomous S ystem
ASN	A utonomous S ystem N umber
AWS	A maزون W eb S ervices
BGP	B order G ateway P rotocol
CA	C ertificate A uthority
CRL	C ertificate R evocation L ist
DDoS	D istributed D enial of S ervice
DNS	D omain N ame S ystem
DNSSEC	D omain N ame S ystem SEC urity E xtensions
DRPKI	D istributed R esource P ublic K ey I nfrasturcture
DS	D elegation S igner
DSA	D igital S ignature A lgorithm
ECDSA	E lliptic C urve D igital S ignature A lgorithm
EU	E uropean U nion
GDPR	G eneral D ata P rotection R egulation
GMW	G oldreich M icali W igderson
GRR	G arbled R ow- R eduction
HSM	H ardware S esurity M odule
INR	I nternet N umber R esource
IP	I nternet P rotocol
ISP	I nternet S ervice P rovider
KSK	K ey S igning K ey
LACNIC	L atin A merica and C aribbean N etwork I nformation C entre
LAN	L ocal A rea N etwork

MAC	M essage A uthentication C ode
MPC	Secure M ulti- P arty Secure C omputation
MP-SPDZ	M ulti- P rotocol SPDZ
NS	N ame S erver
OT	O blivious T ransfer
PKI	P ublic K ey I nfrastructure
PRF	P seudo R andom F unction
RAM	R andom A ccess M emory
RFC	R equst F or C omments
RIPE NCC	R éseaux I P E uropéens N etwork C oordination C entre
RIR	R egional I nternet R egistry
ROA	R oute O rigin A uthorization
ROV	R oute O rigin V alidation
RPKI	R esource P ublic K ey I nfrastructure
RR	R esource R ecord
RRSIG	R esource R ecord SIG nature
RSA	R ivest S hamir A dleman
RTT	R ound T rip T ime
SABUL	S imple A vailable B andwidth U tilization L ibrary
SHA	S ecure H ash A lgorithm
TA	T rust A ncor
TAL	T rust A ncor L ocator
TCP	T ransmission C ontrol P rotocol
TLD	T op- L evel D omain
TLS	T ransport L ayer S ecurity
UDP	U ser D atagram P rotocol
UDT	U DP-based D ata T ransfer
WAN	W ide A rea N etwork
ZSK	Z one S igning K ey

Introduction

The Internet has become an indispensable tool that has enabled us to access information, connect with people around the world and to express ourselves through various media. The opportunities that the Internet creates has often been discussed in terms of the applications that run on top of the Internet infrastructure. Although this infrastructure seldom gets talked about in the mainstream discourse, it is critical to the functioning of the applications. These applications are built on the foundation of the Internet infrastructure. A weak and insecure foundation can be dangerous.

The Internet infrastructure was not designed with security in mind. Fundamental protocols of the Internet such as domain name system (DNS) and routing were designed during a time when security was not a concern. Over the years, attacks on the Internet infrastructure have grown and this has prompted the design of security measures such as DNS security extensions (DNSSEC) and resource public key infrastructure (RPKI). Unfortunately, the deployment of these security measures has been slow and they have failed to deliver on the promised security guarantees. DNSSEC and RPKI rely on cryptographic signatures. Although the private keys should be held by the domain owners and Internet number resource owners, in practice, they are outsourced to centralized authorities.

Centralization of the Internet infrastructure can be harmful in numerous ways. Certificate authorities can be taken over by rogue actors [Pri11], large DNS operators can come under distributed denial of service attacks [DYN16] or state actors can attempt to coerce them to behave against the best interests of citizens [ICA14]. Centralized services are attractive targets for attackers. A single point of trust can turn into a single point of failure.

In this thesis, we address the concerns that arise from the outsourcing of private material and computation to centralized authorities that run the Internet infrastructure. We observe that, in some cases, an intrusion may not only impinge on the security and privacy of individuals, but may affect a group of people and their Internet resource. When we secure a system, we consider a wide range of adversaries. Threats range from nosy corporate employees and hacked databases to government coercion of centralised authorities that manage Internet infrastructure. We build systems that address these threats based on a cryptographic tool known as secure multi-party computation (MPC) or secure computation.

1.1 Secure Multi-party Computation

Secure multi-party computation [Yao86, GMW87] allows multiple parties with private inputs to perform a joint computation and obtain the correct output without having to rely on a trusted third party. Let n participating parties have $x_1, x_2, x_3, \dots, x_n$ as their private inputs. They compute some function f to obtain the desired output $y = f(x_1, x_2, x_3, \dots, x_n)$, such that nothing beyond the output is revealed to the participating parties (Figure 1.1). MPC protocols have to guarantee privacy and correctness. The privacy property enforces that only the necessary information, and nothing else, is revealed to the participants. The correctness property assures us that the malicious behaviour by an adversary does not cause the output of the computation to deviate from that of the intended function.

In MPC protocols, in addition to external adversaries, participating parties are also considered to be potential adversaries. These parties might attempt to learn some private information or to add errors to the computation. Sometimes parties make mistakes while in other cases they might be malicious. Sometimes they might be in control while in other cases they may be threatened by governmental actors to act maliciously. Encompassing such a wide range of adversaries is a feature of MPC that also makes it possible to distribute trust when one or more entities cannot be completely trusted.

While the foundations of computing securely using MPC were laid-down in the 1980s, these protocols are not widely used. While some efficient application-specific protocols have been developed [DKLS18, DKLS19], they do not advance the deployment of MPC as they require cryptographers to design, analyse and prove

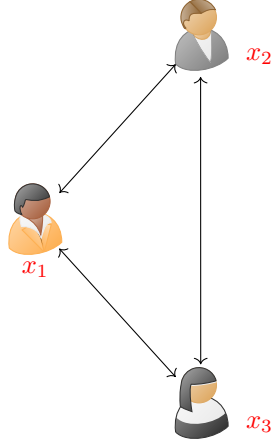


Figure 1.1: Secure multiparty computation.

the security of each of these protocols. Generic protocols can contribute towards a broader deployment of MPC. They can be used in a wide range of applications without considerable assistance from cryptographers as their security only needs to be proven once. This is important when we want to deploy it for the Internet infrastructure. Hence, in this thesis, we focus on generic protocols.

In the context of MPC, we use a fine notion of trust throughout this thesis. When we say that parties are not entirely or completely trusted, we consider them to be trustworthy enough to provide inputs and to process them. However, they are not trusted enough for other parties to reveal their private inputs to them. In the case of securing private keys, while the parties are able to perform the signing collaboratively, they are not trusted with the private key and they cannot misuse it, lose it or leak it.

1.2 Contributions

In this thesis, the focus is on generic protocols, improving their efficiency for practical deployment and using them to secure Internet infrastructure. We ask the following research question:

How can we improve the efficiency of generic MPC protocols that are secure against a diverse set of adversaries such that they can be used to secure the Internet infrastructure?

We answer this research question through the following contributions:

1. **Practical efficiency:** When we consider the use of cryptographic protocols for the Internet, efficiency is one of the primary topics of discussion. Efficiency of MPC should be gauged based on the performance evaluations over the Internet. Systems designed using MPC to secure Internet infrastructure need to have low overhead such that the experience is seamless for end-users. However, until recently, it was hard to analyse the practical efficiency of MPC protocols as many of them have been evaluated on a single machine.

Realistic evaluations are important to understand the efficiency of the protocols over the Internet. We observe in the case of secure two-party computation (2PC), the two party variant of MPC, that the bandwidth is not utilized for functions with large circuits. We observe that the use of different transport layer protocols can improve practical performance. We develop a modular framework, Transputation, that automates transport layer selection and integrates with 2PC implementations. We use the framework to perform extensive evaluations in various network settings to show the benefit of using different transport layer protocols for 2PC.

2. **Efficient threshold signatures:** While cryptography adds overhead to computations, in some cases, most of the cryptographic operations can be pre-processed, before the private information is available. This aspect is true for threshold signatures, a specific instance of MPC. Threshold signatures distribute trust such that no centralized party has control of the complete key and they can only generate signatures in coordination with other parties. Each of them has a “share” of a key, and a signature can be generated only when at least a threshold number of them participate in the signing process. This solution prevents anyone from accessing the signing keys while avoiding reliance on individual authorities.

We design efficient threshold signature protocols based on generic MPC. We propose a transformation that turns any MPC protocol into an equally secure and efficient protocol that computes signatures in a threshold setting. We propose threshold signature protocols in the preprocessing setting. Our protocols have a fast “online” phase where hundreds of signatures can be generated per second.

3. **Securing DNSSEC keys:** With DNSSEC, machines querying a DNS server can verify that the response to a query originates from the correct server and that the response remains unaltered in transit. While DNSSEC is beneficial in principle, its deployment has created new problems. Domain owners typically outsource operations related to the DNS and benefit from increased zone availability and decreased misconfiguration probability. However, with DNSSEC, the handling of private keys is also outsourced. Access to signing keys of domains allows DNS operators to sign bogus records at will. Possession of signing keys of numerous domains makes DNS operators attractive targets for attackers. There is also evidence of poor practices where DNS operators use the same keys for multiple domains.

We address the issue of key management and of centralization of DNS operation by developing a multi-party zone signing system that relies on threshold signatures. Our system is designed for a wide-range of threat models. We implement the system into a widely used DNS software and demonstrate through evaluations that the overhead is minimal compared to traditional DNSSEC when two or three operators are involved. We also perform a measurement study to estimate the number of domains that can already use our system.

4. **Distributed RPKI:** RPKI uses cryptographic signatures that autonomous systems use to validate whether the source of a route is allowed to update routes for the specific IP prefix. The centralized authorities, regional Internet registries (RIRs), that are responsible for allocating IP address space are at the top of the RPKI hierarchy, are completely trusted and the RPKI threat model does not consider the possibility of RIRs being malicious or them being coerced by nation-states. RPKI makes it possible for centralized authorities to takedown IP prefixes.

We propose a change to RPKI that will strengthen the threat model and prevent unilateral takedown of IP prefixes by RIRs. We replace trust in RIRs by designing a distributed RPKI that uses MPC to distribute trust. Our design is automated and can facilitate the deployment of RPKI while reducing or eliminating errors due to manual configurations. We also do not require any changes at relying parties, which makes it easier to deploy our solution.

1.3 Thesis Outline

This dissertation is structured as follows: Chapters 2 and 3 includes the necessary preliminaries and background information for the rest of the thesis. Chapter 2 introduces adversarial models, basics of secure computation and threshold signatures. Chapter 3 introduces network protocols including transport layer protocols, domain name system and routing. Chapter 4 describes the contribution of this thesis towards practical efficiency of 2PC. Chapter 5 describes the development of efficient threshold signatures and how we use threshold signatures to secure DNSSEC keys. Chapter 6 describes the contribution of this thesis towards the design of distributed RPKI system. Finally, chapter 7 concludes this thesis and provides an outlook to potential future research directions.

Secure Computation

This chapter introduces the background information on secure computation that is useful to understand the rest of the thesis. First, we discuss the adversarial models in § 2.1. Then, we discuss a primitive, oblivious transfer (OT), in § 2.2 before we describe two constructions of secure computation protocols in § 2.3. Finally, we discuss the basics of threshold signatures in § 2.4.

2.1 Adversarial Models

In secure computation, protocols are designed based on specific assumptions about the adversaries. First, we consider the behaviour of the adversaries. We assume the capabilities of the adversaries when we design protocols. Second, we consider the number of parties that can be assumed to be corrupt for the protocol to remain secure. Based on these parameters, the efficiency of the protocols can differ.

Adversarial behaviour

With regard to the behaviour of the adversaries, we can design protocols to be secure against adversaries with a range of capabilities. While we refer to Goldreich [Gol04, § 7.2] for formal definitions, we briefly describe the three most commonly used adversarial notions.

1. *Semi-honest* adversaries, also known as *passive* adversaries or *honest-but-curious* adversaries, follow the protocol but can try to extract information from the transcript of the protocol. This adversarial model protects informa-

tion when the parties cannot share plaintext due to legal reasons and when protection against internal adversaries such as nosy employees is required. The design of protocols secure against semi-honest adversaries is the stepping stone towards protocols with stronger security guarantees and they are useful to understand the efficiency of the protocols.

2. *Malicious* adversaries, also known as *active* adversaries, can deviate from the protocol specifications and can try to learn private inputs and influence the outcome of the computation. These are the strongest type of adversaries and they are expensive to protect against as extra checks are needed to ensure that computations are performed correctly.
3. *Covert* adversaries may deviate from the protocol but with the restriction of being caught with a certain probability. The probability of being caught should be high enough to deter attempts by adversaries to act maliciously.

In this thesis, we focus on protocols that are practical in the presence of semi-honest and malicious adversaries.

Number of adversaries

Let us consider protocols where n parties participate. We assume that a minimum number of parties are honest and the rest might be corrupted. The protocols where a majority of the parties are assumed to be honest are called *honest majority* protocols. The less restrictive case where it is not necessary for the majority to be honest for the protocol to be secure is known as *dishonest majority* protocols. Honest majority protocols are typically more efficient. In the special case of two-party computation, we can only assume at most one party to be corrupt, and hence, the notion of honest majority does not arise. In the multi-party case, we include protocols in both honest and dishonest majority settings.

2.2 Oblivious Transfer

Oblivious transfer [Rab05] is a cryptographic primitive in which one party, a sender, transmits multiple messages to another party, a chooser, such that the sender does not learn which message has been received by the chooser while the chooser does not

learn anything about the other messages. In a 1-out-of-2 OT, the sender inputs two messages, say m_0 and m_1 , and the chooser inputs a choice bit $b \in \{0, 1\}$. At the end of the protocol, the chooser obtains m_b , being ignorant of m_{1-b} , while the sender gains no information. Impagliazzo and Rudich [IR89] showed that public key cryptography is necessary for OT and that they cannot be constructed from symmetric key cryptography. Therefore, OT protocols must be instantiated based on public-key type assumptions such as RSA or the Decisional-Diffie Hellman assumption. However, in practice, the high computational cost of exponentiations required by OT protocols can be reduced through the use OT-extension technique [Bea96, IKNP03], which makes it possible to generate an unbounded number of OT, using only symmetric crypto operations, from a few base OTs.

2.3 Secure Computation Constructions

Secure computation allows multiple parties to jointly compute any function of their inputs and obtain the result without leaking any other information. The seminal results from the 1980s [GMW87, Yao86] showed that it is possible to evaluate any function in a secure way. That is, n parties P_1, P_2, \dots, P_n with inputs x_1, x_2, \dots, x_n can jointly evaluate some function f on their inputs in such a way that these parties learn the desired output $y = f(x_1, x_2, \dots, x_n)$ and *nothing else* about the input of the other parties.

In secure computation, the function f is converted into a circuit and the protocol is run on that circuit. The function f is expressed as a boolean or an arithmetic circuit. A boolean circuit works on bit inputs, and is represented by a combination of AND and XOR gates while an arithmetic circuit works on inputs that are elements of some field \mathbb{F} , and is represented by a combination of addition and multiplication gates. Hence, secure computation protocols are constructed as a combination of addition and multiplication protocols.

There are two main types of constructions that are considered in this thesis: Yao's garbled circuits for 2PC in § 2.3.1 and secret sharing for MPC in § 2.3.2.

2.3.1 2PC Based on Garbled Circuits

In Yao's garbled circuit protocol [Yao86], two parties interactively compute over a garbled version of a boolean circuit. One party, say P_1 , chooses two random keys

for every wire in the circuit and then uses them to construct garbled gates. For each wire i , two random keys are sampled k_i^0 and k_i^1 ; one that logically represents the plaintext 0 bit and one that represents the plaintext 1 bit. Then, P_1 uses these keys to construct garbled gates, wherein a boolean gate g has input wires u, v and output a wire w . P_1 constructs a table of ciphertexts consisting of encryptions of the output keys under the corresponding input keys, i.e., a garbled gate is constructed by concatenating ciphertexts of the form

$$C_{a,b} = E(k_u^a, k_v^b; k_w^{g(a,b)})$$

for all four combinations of $a, b \in \{0, 1\}$.

P_2 then obviously evaluates the garbled gates, i.e., P_2 will learn one key for each input k_u^a, k_v^b but not the values a, b . Nevertheless, P_2 will be able to learn the corresponding output key $k_w^{g(a,b)}$, but neither $g(a, b)$ nor $k_w^{1-g(a,b)}$. The ciphertexts in the garbled gate are permuted at random to ensure that P_2 learns nothing about the output value $g(a, b)$. Therefore, P_2 must decrypt all 4 ciphertexts and have a mechanism to identify the right key. This is achieved by adding redundancy to the plaintexts that in turn increases the communication complexity of the protocol.

To allow P_2 to begin the evaluation, the keys corresponding to the input wires must be sent from P_1 to P_2 . The input wires belonging to P_1 are sent directly while those belonging to P_2 are sent obviously through OT. Then P_2 evaluates the circuit, non-interactively, gate by gate and obtains an encrypted output. Finally, if P_2 is supposed to learn the output value, P_1 also sends some decoding information which allows P_2 to decode the keys corresponding to the output wires to their plaintext values. Because of their roles in Yao's garbled circuit protocol, P_1 is also known as the *Garbler* and P_2 as the *Evaluator*.

Yao's protocol was sub-optimal in terms of computation and communication. To improve the efficiency of Yao's protocol, numerous computation and communication optimizations such as point-and-permute [BMR90], garbled row reduction [NPS99, PSSW09], XOR-1 [GLNP15], free-XOR [KS08], fixed-key AES [BHKR13], half-gates method [ZRE15] have been proposed. We discuss them in more detail in Chapter 4.

We have described Yao's garbled circuit protocol that is secure in the semi-honest security model, which is the focus of Chapter 4. The security proof of Yao's garbled circuits protocol can be found in Lindell and Pinkas [LP09]. We note that garbled circuits are only secure for one-time use. Reusable garbled circuits can be designed to be secure, but they are not yet practical [GKP⁺13]. It is also theoretically possible

to garble arithmetic circuits [AIK14]. Yao’s protocol has also been modified to be secure against malicious adversaries [LP07, LR15].

2.3.2 Secret Sharing

Secret sharing is a cryptographic tool that is at the heart of many MPC protocols. It is a tool to spread information about a secret among many parties such that together they hold the complete information on the secret, but individually no party has the secret. One form of secret sharing is *additive secret sharing*. Consider a secret $s \in \mathbb{Z}_p$, where $\mathbb{Z}_p = \{0, 1, \dots, p-1\}$ is a finite field for a prime number p . The secret s is shared among n parties P_1, P_2, \dots, P_n by sending x_1, x_2, \dots, x_n such that $x_i \in \mathbb{Z}_p$ and $\sum_i x_i = s \bmod p$. The x_i s are known as the *shares* of the secret s and any missing shares will prevent the reconstruction of the secret.

While we use additive sharing scheme for dishonest majority protocols in Chapter 5 and 6, for honest majority protocols, we rely on *Shamir’s secret sharing* and *replicated secret sharing*. In Shamir’s secret sharing [Sha79], the secret is shared by choosing a random polynomial and determining the shares using this polynomial. The degree of this polynomial is at most the maximum number of corrupted parties. Replicated secret sharing [BL88] is similar to additive secret sharing, but instead of one share, every party receives multiple shares. Both these schemes make it possible to recover the secret as long as a certain threshold number of shares are available. All these secret sharing schemes—additive, Shamir’s, and replicated—are linear, i.e., given the shares, the secret can be reconstructed through a linear combination computed locally.

At a high-level, an MPC protocol based on secret sharing takes the following form:

- Assuming the parties have agreed on an arithmetic circuit to compute jointly, the parties share their secret-shared inputs with each other.
- They compute the arithmetic circuit by performing additions and multiplications over secret shared values.
- In the final step, the parties ‘open’ or reveal the result of circuit evaluation.

To compute over arithmetic circuits we have to show how the shares are added and multiplied. Let $[x]$ denote a secret sharing of x such that each party P_i holds

the share $[x_i]$. Scalar addition and multiplication can be computed locally: $a + [x]$ and $a \cdot [x]$. We now consider addition and multiplication of secret shared values.

Addition. To add x and y when the parties hold the shares $[x], [y]$, we locally add the shares:

$$[x + y] = [x] + [y].$$

Multiplication. To multiply x and y , Beaver [Bea91] proposed the method of *circuit randomization* to securely multiply random values and then use these to multiply actual values for secure computation. Let us take one-time-use *multiplication triples* of the form $(a, b, c) \in \mathbb{Z}_p$ such that $c = ab$. Each party holds a share of the triple $([a], [b], [c])$ and uses it for multiplication. Given the shares $[x], [y]$ and a multiplication triple $([a], [b], [c])$, Beaver's technique is as follows:

1. Each party locally computes and broadcasts the following values:

$$[e] = [a] + [x]$$

$$[d] = [b] + [y].$$

2. Each party opens e and d , such that $e = a + x$ and $d = b + y$. As each party only knows the shares of the multiplication triple that are uniformly random, the shares of x and y are hidden by the triples.
3. The parties can then locally compute their share $[z]$:

$$[z] = [c] + e[y] + [x]d - ed.$$

We can show that the share $[z]$ corresponds to the share of the product xy :

$$\begin{aligned} z &= c + ey + xd - ed \\ &= c + (a + x)y + x(b + y) - (a + x)(b + y) \\ &= c - (a + x)b + x(b + y) \\ &= c - ab + xy \\ &= xy. \end{aligned}$$

Circuit randomization allows us to push a lot of the computation to the pre-processing phase so that we can have an efficient online phase. The preprocessing can be batched to perform many multiplications on random data. This results in the overall protocol being more efficient. We will use this technique in Chapter 5. Additional information on secret sharing can be found in Cramer, Damgård and Nielsen [CDN15].

2.4 Threshold Signatures

A digital signature scheme consists of three operations—key generation, signature creation and signature verification—that are represented by a tuple $(\text{KGen}, \text{Sig}, \text{Vf})$ where:

- $\text{KGen}(1^\lambda)$ on input a security parameter 1^λ , outputs a key pair (sk, pk) where the signing key sk is used to create signatures and is kept secret, and the public key pk is used for verification and made public.
- $\text{Sig}(\text{sk}, M)$ on input the signing key sk and message $M \in \{0, 1\}^*$, produces a signature σ .
- $\text{Vf}(\text{pk}, M, \sigma)$ on input the public key or verification key pk , message M and signature σ , outputs 1 if σ is a valid signature on M and 0 otherwise.

A digital signature scheme satisfies two properties:

Correctness. With a high probability, all valid signatures should be verified. Given $(\text{sk}, \text{pk}) \leftarrow \text{KGen}(1^\lambda)$, any message M signed using the secret key sk should be verified using the public key pk with a high probability.

Existential unforgeability. An adversary should not be able to forge a signature σ on a message M with greater than negligible probability if a legitimate signer has not signed the message M before.

Threshold signatures, an idea introduced by Desmedt [Des87], allow a group to delegate their authority to a subcommittee to sign a message. Specifically, threshold signatures enables a group of n parties to jointly generate a single public key pk and then compute a signature only if a certain threshold, say $t + 1 \leq n$, participate in the signing protocol. In addition to the two properties of a digital signature scheme,

threshold signatures must also satisfy the privacy property, that is, any group of t or less parties will not learn anything about the signing key being used. They will neither be able to generate a valid signature on a previously unsigned message nor will they be able to deceive an honest party to unwillingly sign a message. Threshold signatures can be thought of as a specific instance of MPC.

The appeal of threshold signatures is that they distribute the signing process and they make it more robust to adversarial attacks or disruptions, while preserving the original verification procedure of the signature scheme. In particular, signatures that are generated by the threshold protocol can be verified as if they were computed by a single party. The first threshold signature scheme was proposed by Desmedt and Frankel [DF89]. This work required a trusted third party to distribute the keys. This requirement was removed in the work of Pederson [Ped91]. Through the 1990s and early 2000s, threshold signature protocols for DSA [GJKR96, MR01], RSA [Sho00, DK01] and Schnorr [SS01] signatures have been proposed.

In the recent years, there has been a resurgence of threshold signatures, specifically, for ECDSA [Lin17, LNR18, GG18, DKLS18, DKLS19]. This renewed interest can largely be attributed to their role in crypto-currency schemes where security of the signing key is paramount as it is used to authorize transactions; in a nutshell, the key *is* the account. Therefore, the possibility to increase the security of the key by splitting the role of the signer among several separate entities, without changing the verification procedure, makes it attractive to use threshold signatures in crypto-currency schemes.

Network Protocols

This chapter provides background information on network protocols that are relevant to this thesis. First, we discuss transport layer protocols in § 3.1. Then, we discuss domain name system security extensions in § 3.2 before we discuss border gateway protocol and resource public key infrastructure in § 3.3.

3.1 Transport Layer Protocols

Transmission control protocol (TCP) exhibits unsatisfactory performance on inter-continental links with high bandwidth delay product and links with high packet loss rates. This performance degradation may not be significant if the amount of data transferred is low. In applications with megabytes of data being transferred, this performance degradation can be critical and yet TCP is still being used. Though approaches to improve TCP’s congestion control have been proposed [HRX08, CF04, WJLH06, WJLH06, LBS08, WFGZ10, CCG⁺16], they fail to perform consistently when the network assumptions on which they are based is violated.

Most congestion control algorithms rely on indicators such as packet loss and delay to determine the congestion in the network. TCP-CUBIC [HRX08], used by default in Linux kernels since version 2.6.19, is a loss-based congestion control algorithm in which the congestion window is a cubic function of time since the last packet loss. It fills the available buffer capacity leading to large queueing delay. Bottleneck Bandwidth and Round-trip propagation time [CCG⁺16], a recently proposed congestion control algorithm, does not rely on loss or delay indicators but

estimates the available bandwidth to determine the sending rate and tries to avoid queueing delay. This is still a work in progress and in its current state, issues such as increased queueing delay and massive packet losses have been observed [HBZ17].

There are alternatives to TCP such as Simple Available Bandwidth Utilization Library (SABUL) [GG03, GG07, GG08], QUIC [QUI] and Performance-oriented Congestion Control [DLZ⁺15]. SABUL and its variant UDP-based Data transfer protocol (UDT) are UDP-based protocols with the goal of utilizing the bandwidth efficiently. Congestion control and reliability control mechanisms of SABUL are built on top of UDP in the application layer, making it easy for use by application developers. It was designed for transferring large data streams over high-speed wide area networks but has since been updated to support commodity Internet.

3.1.1 TCP-CUBIC

TCP-CUBIC uses a cubic function to manage the congestion window size. It uses a convex component as well as a concave component for increasing the window size. After window reduction on packet loss, the concave component contributes to the rapid increase in the window size. As it nears the window size W_{max} before the packet loss, the growth slows down to zero. Then TCP-CUBIC begins looking for more bandwidth and the window grows slowly away from the previous W_{max} due to the convex component. Considerable time is spent between the concave and convex growth regions as the network stabilizes to probe for more bandwidth. The congestion window of TCP-CUBIC is given by the following equation:

$$W = C(t - K)^3 + W_{max}$$

where C is a scaling factor, t is the elapsed time from the last window reduction, W_{max} is the window size before the last packet loss, and $K = \sqrt[3]{W_{max}\beta/C}$, where β is a constant multiplication decrease factor applied for window reduction at the time of loss event. In the rest of the thesis, we refer to TCP-CUBIC as TCP.

3.1.2 SABUL

SABUL adds reliability at the application layer on top of UDP. It makes use of timer-based selective ACK and packet-based sequencing. Packet loss is indicated through the use of a negative acknowledgement. It makes use of a hybrid rate-based

congestion control and window-based flow control. Rate control, which manages packet sending rate, is triggered at every constant interval (*SYN*) while window control, which limits the number of unacknowledged packets, is triggered when an acknowledgement packet is received. Packet sending rate is controlled through an additive increase and multiplicative decrease algorithm. Multiplicative decrease is by a factor of 1/9, while additive increase is independent of the round-trip time (RTT). Additive increase factor is given by the following equation:

$$\alpha(x) = 10^{\lceil \log(L-C(x)) \rceil - \tau} \cdot \frac{1500}{S} \cdot \frac{1}{SYN}$$

where x has the unit of packets/second. L is the link capacity measured by bit-s/second. S is the SABUL packet size (in terms of IP payload) in bytes. $C(x)$ is a function that converts the unit of the current sending rate x from packets/second to bits/second ($C(x) = x * S * 8$). $\tau = 9$ is a protocol parameter.

3.2 Domain Name System

The domain name system [Moc87a, Moc87b] is one of the core-infrastructures of Internet. DNS resolves human-readable domain names to machine-readable Internet protocol (IP) addresses over the Internet. The basic unit of DNS organization is the *zone*. A domain such as **example.com** is a zone, which contains records about the hosts and child zones. The zone **.com** contains name server **NS** record for the child zone **example.com**. The zone file for the domain **example.com** contains IPv4 address **A** record, IPv6 address **AAAA** records for the hosts, mail exchange **MX** records, among others. However, DNS lacked security features, making it vulnerable to numerous attacks such as DNS cache poisoning and DNS hijacking [Bel95, AA04, Kam08, SS10, HS13b].

3.2.1 Domain Name System Security Extensions

DNSSEC [AAL⁺05a, AAL⁺05c, AAL⁺05b] secures the information in zone files by cryptographically signing the records so that an intruder who wants to insert bogus data will have to compromise the private key as well as the authoritative name server, which has the responsibility of translating domain names to IP addresses for the domains under its control. DNSSEC provides data integrity but not availability

and confidentiality. DNSSEC defines new resource records (RRs) to store keys and signatures which are used to authenticate DNS responses:

DNSKEY. These are public keys whose corresponding private key is used to sign RRsets. DNSKEY is used to verify these signatures. Each zone usually creates two DNSKEY records—one for a Key Signing Key (KSK) and another for a Zone Signing Key (ZSK). The private key of the KSK is used to sign DNSKEY records, and the private key of the ZSK is used to sign all other records.

RRSIG (Resource Record Signature). The RRs of the same type and name in a DNS zone are grouped to form RRsets and the RRsets in a zone file are hashed and signed using the private key corresponding to the DNSKEY.

DS (Delegation Signer). Contains the hash of the public key (DNSKEY) of the child zone. It is uploaded to the parent zone by the registrar. The DS records are signed by the parent zone. It establishes a chain of trust between the parent and the child zone.

3.2.2 Organizations in DNS signing setup

Registries are organizations that manage top-level domains (TLDs). They are responsible for maintaining TLD zone file.

Registrars are organizations that sell domains to the public. When a registrar sells a domain name, it updates the registry with information such as NS record and DS record (if DNSSEC is supported).

Domain owner is the owner of the domain name and is the customer of the registrar.

DNS operators are organizations that run authoritative name servers. Each domain name has at least one operator while an operator can serve as the DNS server of many domains.

3.2.3 Zone Signing

Zone signing is an important component of DNSSEC. The resource records of the same type are grouped to form RRsets before they are signed. These signatures and

the `DNSKEY` (the public key corresponding to the secret key used to sign) are used by security-aware DNS clients to verify the validity of DNS data. Signing is usually performed by organizations operating the name servers for the domain. These organizations could be DNS hosting provider, web hosting provider or a registrar. Note that a zone file needs to be re-signed in the following situations:

1. The RRs are changed.
2. The signatures need to be refreshed.
3. The keys are changed or rolled over.

DNSSEC uses two pairs of keys: KSK and ZSK. The private key of KSK is used to sign `DNSKEY` records while the private key of ZSK is used to sign the rest of the RRsets. The reason for having two pairs of keys is that the key used to sign the RRsets needs to be changed at regular intervals to prevent attackers from guessing the private key. When the ZSK is changed, the registrar does not need to be involved (if the registrar is not the DNS operator). Rolling over a KSK involves publishing a new `DNSKEY` and updating the `DS` in the parent zone, which involves the registrar. As KSK is recommended to be changed once an year and is kept offline while ZSK is changed much more regularly, using two-pairs of keys reduces the overhead and safeguards the secret keys. Note that DNSSEC does not have a key revocation mechanism. Keys are supposed to be rolled-over periodically and if the keys are compromised. Details of key rollover and the recommended best practices can be found in RFC 6781 [KMG12].

The mode of zone signing may depend on the frequency of changes in the zone and the DNS operator. Depending on the domain, the zone may be static or dynamic. A dynamic zone can be updated if the records in the zone file changes. For example, if the IP address for the domain changes, then the zone file needs to be updated. A static zone, on the other hand, is a complete set of records that is created and signed offline. Whenever records in the zone are updated, the corresponding RRsets need to be signed.

DNSSEC signing can take two forms: offline signing and “on-the-fly” signing. While offline signing is sufficient for static zones, many operators use “on-the-fly” signing, where the RRsets are signed after a DNS query is received. The signature is generated “on-the-fly” and sent with the DNS response. The efficiency of the signing system becomes more important in the case of “on-the-fly” signing.

3.2.4 Registrar as DNS Operator

Some of the registrars offer their customers the option where (1) the domain owner can operate the domain, or (2) the registrar becomes the DNS operator and serves as the authoritative name server for the domain being purchased. With regard to DNSSEC, the registrar plays a critical role irrespective of the option chosen by the domain owner. The registrar registers its customer's domain name with the registry, updates the `NS` record as well as `DS` record. Typically, only the registrar is allowed to modify information about the domain names it has registered at the registry. As only the registrar can upload a `DS` record for the domain to the registry and a missing `DS` breaks the chain of trust, the registrar plays a critical role in the DNSSEC ecosystem. In Chapter 5, we also consider the scenario where the registrar is the DNS operator. In this setting, the registrar can directly upload the `DS` record when it is available. However, we are concerned about the registrar (as the domain operator) having complete control over the domain, especially the private keys.

3.2.5 Multi-Operator Setting

The evolution of DNS over the past two decades has seen the move from the recommendation to swap secondary zones with other organizations [EBBP97] to the reliance on DNS operators with global points of presence. Nevertheless, many domains were not accessible in the face of large-scale distributed denial of service attacks, e.g., on NS1 [Bee16] and Dyn [DYN16] in 2016, because of their reliance on a single DNS operator. One solution to safeguard against such attacks on the infrastructure of a single operator is to use multiple DNS operators to host zone files. If the zone is signed by one signing server and only served by the other operators as a secondary authoritative name server, then standard zone transfer techniques suffice. However, some operators only support online signing while others offer non-standard DNS features. DNSSEC in this setting is challenging [HAD⁺20].

3.3 Routing

Internet infrastructure relies on the border gateway protocol (BGP) to perform inter-domain routing between autonomous systems (ASes). An AS is identified by a unique number, known as AS number (ASN), that is assigned to the AS along with IP prefixes by one of the five RIRs. ASes announce the path to a destination IP

prefix to establish routing tables. This announcement, known as BGP announcement, includes the destination IP prefix (e.g., 192.0.2.0/24) and the ASNs on the path. When other routers receive such announcements, they update their routing tables. Usually, they choose the most specific prefix when making routing decisions.

As the original BGP did not have security features, it is vulnerable to attacks. A malicious AS could announce a route for a prefix that it does not own, which may not only result in traffic interruption but can also be used to intercept traffic. Such an attack is known as *prefix hijacking*. A malicious AS may also announce a more specific prefix (say, 203.0.113.192/26) than the originator's prefix (203.0.113.0/24). As routers usually choose the more specific prefix to route traffic, the packets will be forwarded to the malicious AS. Such an attack is known as *sub-prefix attack*.

3.3.1 Resource Public Key Infrastructure

Resource public key infrastructure is an out-of-band mechanism that uses cryptographic signatures to prevent prefix and sub-prefix attacks. The signatures are used to specify the entities that are authorized to announce specific prefixes. RPKI architecture includes trust anchors, certificate authority (CA) certificates and end-entity (EE) certificates. A *trust anchor* is a self-signed X.509 CA certificate [LKS04, CSF⁺08] that is at the head of the chain and is assumed to be trusted. Each RIR has a trust anchor. In X.509 architecture, the chain of trust is derived from this authoritative certificate. The trust anchor contains a public key in the `subjectPublicKeyInfo` field along with the associated data that are used when a signature on a certificate or a signed object is validated [RW10, HWM⁺19]. A trust anchor is used to sign the CA certificates of each member.

A resource holder needs a CA certificate to sub-allocate resources and to issue resource certificates. A CA certificate binds Internet Number Resources (INRs) and IP prefixes to a public key. CA certificate is used to sign EE certificate, which verifies signed objects. The private key corresponding to the public key in an EE certificate cannot be used to sign other certificates. There is a one-to-one mapping between EE certificate and signed objects. If the EE certificate is revoked, then the corresponding signed objects are automatically revoked.

Route Origin Authorizations (ROAs) are digitally signed objects, X.509 certificates, that provide a method to verify that an IP address block holder has authorized an AS to originate routes to specific prefixes within that address block (Figure 3.1).

Prefix	203.0.113.0/24
MaxLength	/24
Origin AS	AS 12345

Figure 3.1: Route origin authorization.

While each ROA includes exactly one ASN, multiple ASNs may be authorized, with each ASN requiring a separate ROA. Each ROA, though, can contain multiple IP prefixes with their prefix lengths.

A Certificate Revocation List (CRL) is a list of resource certificates that have been revoked, and should not be relied upon by the relying parties. A CRL is always issued by the same CA that issues the corresponding certificates. All these objects are published in public repositories.

3.3.2 Hosted and Delegated RPKI

There are two RPKI models: delegated RPKI and hosted RPKI. In the delegated RPKI model, AS runs a CA as a child of RIR, generates its own certificate, gets it signed by the parent CA. This model allows the AS to operate independently from the parent RIR. For large operators of a global network, this model is suitable so that they do not need to maintain ROAs through the different web interfaces of the RIRs. However, this model is not suitable for all as it requires running a CA and maintaining the ROAs.

In the hosted-RPKI model, RIRs host the CA, i.e., the same entity that allocates IP resources also runs the CA to validate the ROAs. Thus, in this model, they are trust anchors. In a way, this is meaningful as the RIRs already know the owner of the address space. Existing RPKI systems are tied-up with the login credentials of the ASes at the RIR. Signing and key rollover are automatic. It is easy for the owners of the address space to begin using hosted RPKI than delegated RPKI as the CA functionality is handled by the RIR. This model is convenient for most ASes. Even large providers such as Cloudflare make use of hosted RPKI ¹. Furthermore, the RIR assumes responsibility to publish the signed objects. However, this convenience comes at the cost of further centralization of power as the RIRs also handle the private keys used to sign ROAs.

¹<https://ripe77.ripe.net/presentations/156-RPKI-deployment-at-scale-RIPE-1.pdf>

3.3.3 Route Origin Validation

RPKI has two sides: (1) creating certificates and RPKI objects (2) validating the ROAs of others. So far we have discussed the former. When it comes to validating, the ASes use relying party software to download and validate RPKI objects from one of the public repositories. In the rest of the thesis, We refer to ASes in this capacity as relying parties. When a relying party receives a BGP announcement, it attempts to validate the announcement and assigns one of the following states to the announcement:

- **Valid:** The announcement is covered by at least one ROA such that
 1. the origin ASN in the announcement matches the ASN in a ROA,
 2. the IP prefix in the announcement is covered by a ROA, and
 3. the IP prefix in the announcement is not greater than the MaxLength in the AS.
- **Invalid:** The announced IP prefix is covered by a ROA but the ASN in the announcement does not match the ROA.
- **Unknown:** The announced IP prefix is not covered by an existing ROA.

Optimal Transport Layer for Secure Computation

Secure two-party computation allows any two parties to perform a computation over their private inputs and obtain the correct output of the computation without leaking any other information. 2PC ensures the privacy of the inputs of the participants and of the computation. 2PC eliminates the need for a trusted third party when two non-trusting parties want to obtain an output by computing a function. From a practical point of view, the underlying premise is that, if the two parties cannot trust each other, how probable is it that they will be able to find a mutually trustworthy party to facilitate a computation?

Secure two-party computation has the potential to facilitate numerous applications while maintaining user privacy. 2PC is a powerful tool that enables collaborations that were previously infeasible. Presently, possession of sensitive personal data prevents governments, companies and individuals from engaging in online transactions with each other due to reasons such as lack of trust between them, concern for the privacy of their customers as well as privacy regulations such as the General Data Protection Regulation (GDPR). Examples of such applications include key management for digital currencies [ABL⁺18], auctions [BCD⁺09], tax-fraud detection [BJSV15], private set intersection [CMP11, HEK12, PSZ18], and prevention of satellite collision [KW15, HLOI16].

Nevertheless, 2PC is typically not used in real life applications due to performance issues and it has not yet received widespread acceptance from the industry. The

current compromise is that many transactions (aforementioned and others) either require sacrificing the privacy of users (e.g., via a third party) or cannot be performed online.

Computation is optimal

While the theoretical efficiency of 2PC protocols can be understood in terms of the computation and communication complexity, they are not sufficient to understand the practical efficiency of 2PC protocols. The first public implementation of 2PC known as Fairplay was released in 2004 [MNPS04]. This implementation has been the starting point for the huge progress made in terms of 2PC protocol design and implementation engineering. The most efficient implementations of 2PC protocols are based on the protocol proposed by Yao [Yao86] which is built from garbled circuits [BHR12] and oblivious transfer [NP01]. Over the years, due to protocol optimisations, hardware support for cryptographic operations and the use of multiple cores, the computation overhead of 2PC protocols has been reduced drastically and there are indications that the current constructions of 2PC based on garbled circuits have reached the theoretical lower bound [ZRE15]. It is now widely believed that the *bandwidth*, and not the *computation*, is the remaining bottleneck in 2PC [ALSZ13, ZRE15].

Implementations are still not practical

Since Fairplay, there have been numerous prototype implementations of 2PC protocols [HEKM11, EFLL12, DSZ15, ZE15, WMK16]. Although the number of implementations has increased, it is important to note that most of these implementations have been developed to demonstrate feasibility [Hal18]. Some of the implementations are evaluated in simulated environments or on a single host that neither take into account realistic network conditions nor the presence of other processes [Kre17]. Current 2PC as well as MPC implementations are difficult to use for developers [HHNZ19]. The result is that users have to trade-off their privacy with efficiency by resorting to third parties to perform computations for them instead of running 2PC with the target service.

Communication is yet to be explored

Although there has been work on reducing the communication complexity and limiting the amount of data that needs to be exchanged between the two parties, there has not been any research on the practical communication efficiency of 2PC. Indeed, evaluations of 2PC implementations on a single machine now result in practical performance [BHKR13, SZ13]. Nevertheless, when 2PC is evaluated on separate hosts, the latency overhead is prohibitive for practical applications [NST17, WRK17]. Evaluations of 2PC disregard issues faced by practical deployment of 2PC: many implementations are not evaluated in real life setups where diverse network conditions, such as packet loss, latency, and other traffic can impact performance [Kre17, Hal18].

Contributions

Our contributions are as follows:

Yao vs. GMW. We show evidence through evaluations that 2PC protocol based on Yao’s garbled circuits is more efficient than GMW in all network settings. The prior understanding was that GMW was more efficient in LAN setting. Our evaluations show that this is not the case any more because of the optimizations to garbled circuits. Based on this evidence, we focus on Yao’s protocol in the rest of this chapter.

Network utilisation of secure computation protocols. We study the network utilisation of secure computation implementations. Our evaluations indicate that the bandwidth of secure computation implementations is under-utilised. As in Fairplay [MNPS04], all the implementations use standard TCP for communication irrespective of the network setting in which the scheme is to be deployed [HEKM11, EFL12, DSZ15, ZE15, WMK16]. This is surprising as there has been considerable research towards choosing the right 2PC scheme for a given setting. In contrast to improvements in computation and communication complexity, the transport layer has so far received no attention from the 2PC research community.

Transputation framework. We bridge the gap between research in 2PC protocols and transport layer protocols by developing Transputation framework. Transputation provides an interface for easy and automated integration of

transport layer protocols for secure computation. It is a modular framework that optimizes the practical performance of 2PC protocols by choosing the transport layer protocol based on the parameters of the 2PC protocol and the network conditions. We have integrated three transport layer protocols into Transputation: UDP, TCP and SABUL¹. Given a function to be computed securely and the circuit size, Transputation estimates the network conditions to automatically choose the optimal transport layer protocol in the framework. The modularity of the framework makes it easy to add new transport protocols as well as 2PC implementations independently. The source code implementation of Transputation can be accessed at <https://github.com/Fraunhofer-SIT/transputation>.

Evaluations of 2PC. We use Transputation to perform extensive evaluations. First, we simulate different network conditions to understand the performance of 2PC protocols with different transport layer protocols before we consider the complexity of the Internet and evaluate the performance in large-scale real world networks. Through our evaluations using Transputation, we demonstrate that even general purpose protocols already provide significant performance improvement over standard TCP in certain network conditions. For instance, for large circuit sizes in WAN setting, SABUL performs $7 - 8\times$ better than TCP (Figure 4.1). We note that although special purpose transport protocols could improve the performance further, we focus on general purpose protocols to foster widespread deployment.

Chapter Outline

Section 4.1 introduces Transputation framework and its layers. The optimizations and implementations in secure computation layer and the transport layer are explained in Section 4.2 and 4.3 respectively. Section 4.4 presents the design and implementation of Transputation framework. Section 4.5 reports the simulation and evaluation results of Transputation. Section 4.6 reviews related work.

¹Though we use the most updated version of SABUL, also commonly known as UDT or UDP-based Data Transfer Protocol, we use the former name to stress the difference with “regular” UDP.

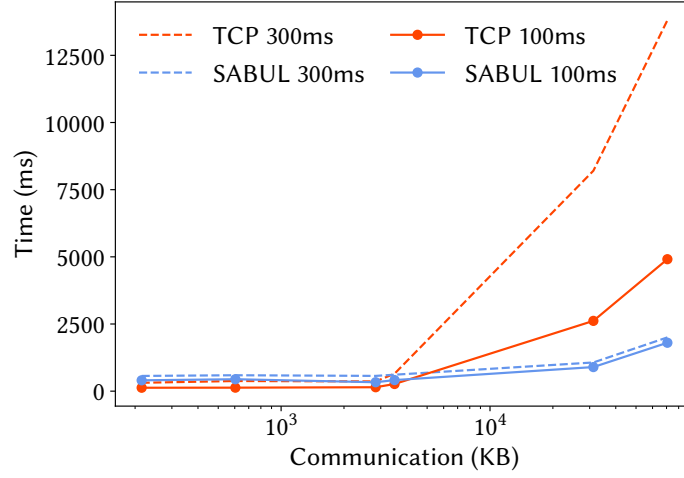


Figure 4.1: TCP vs. SABUL as garbled circuit size increases.

4.1 Transputation Framework

In this section, we provide an overview of Transputation and explain how it can be used. We describe in § 4.1.1 the components of Transputation before we describe in § 4.1.2 how Transputation can be used.

4.1.1 Components

The goal of Transputation is to identify the transport layer protocol that is optimal for a given function to be computed securely in real network settings. To pursue this goal, our framework has two layers: the *secure computation* layer and the *transport* layer (see Figure 4.2).

Secure Computation Layer. This layer is composed of 2PC implementations that accept the function or application to be evaluated securely and the size of the circuit representing the function. In § 4.2, we motivate our choices for 2PC implementations that we integrate into Transputation. We identify the parameters that impact the data volume and communication pattern, and deploy optimizations relevant for performance improvements on the transport layer.

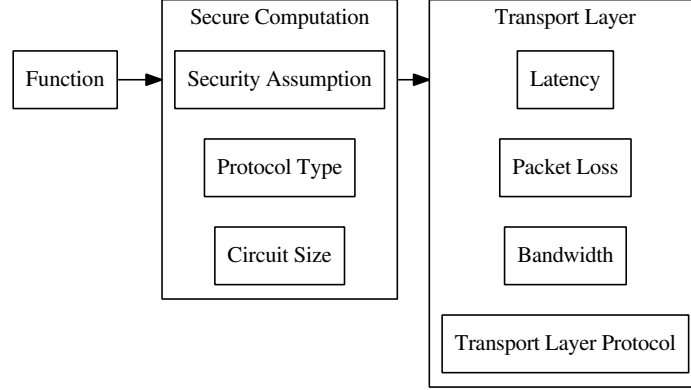


Figure 4.2: Transputation framework.

Transport Layer. This layer is responsible to identify an optimal transport layer protocol for a given computation task. To that end, it determines the latency, bandwidth and packet loss on the network through live experimental probes. Transport layer is also responsible for avoiding overflow at the sender and the receiver. To avoid overflow, given the function to be computed and the size of inputs to the circuit, Transputation calculates the required buffer sizes on the sender and the receiver. We explain the transport layer implementation in § 4.3.

We integrate, in both layers, representative protocols to demonstrate the usage of framework as well as the evaluation of 2PC implementations. Transputation brings together the *secure computation* layer and the *transport* layer by choosing a suitable transport protocol based on the combination of parameters obtained from them. Transputation is built to support easy extensions at both layers. New implementations of 2PC as well as new transport layer protocols can be integrated into Transputation. The flexibility for easy integration of transport layer protocols is critical—new protocols are continually devised, e.g., to improve performance of applications in diverse network conditions or to support new Internet architectures. Future protocols may improve the performance of 2PC more than the existing options. Transputation provides agility and flexibility for 2PC developers enabling them to constantly extend the implementations with new transport protocols.

4.1.2 Usage of Transputation

We have implemented abstract classes into Transputation that simplifies and automates the selection of transport layer protocols. Adding a new transport layer protocol to the framework only requires specifying a string to identify the protocol without requiring any changes to the 2PC implementation. Without the framework, substantial parts of the 2PC implementation will need to be re-written to accommodate new transport layer protocols. In many existing secure computation implementations, the code is not modular. The network code is interwoven with the application code. The transport layer protocol as well as the IP address and the port numbers are hard-coded into the application code. This problem has also been independently observed by Halevi [Hal18]. Unwrapping such an implementation and adding transport layer protocols requires extensive rewriting of the code base. To begin with, an entangled code base prevents retrofitting of transport layer protocols. The code base will need to be made modular before considering the integration of different transport protocols.

Secure computation developers can use Transputation in three ways:

1. Developers can use the *transport layer* module to automatically choose the most appropriate transport protocol in a given network setting for their 2PC implementation and application. Developers can implement the 2PC protocol without worrying about the underlying transport layer protocol. Then, the developers can invoke the *transport layer* part of Transputation that is implemented as a wrapper. Our implementation provides functions to set up a connection between the two parties, to send and to receive data. The 2PC implementation is then integrated with the transport layer protocol that provides the most efficient run time.
2. Developers can also test the 2PC protocol manually by experimenting with the various transport layer protocols available in Transputation under various network conditions for a particular 2PC protocol implementation. This allows developers to choose the most suitable transport layer protocol for the 2PC application.
3. Secure computation protocols consist of interactive primitives such as OT and OT extensions. Transputation can be used not only by developers who use

these primitives to construct a 2PC protocols, but also by the designers of the primitives to test the practical efficiency of their design.

4.2 Secure Computation Layer

In this section, we present the secure computation layer of Transputation. We compare the performance of 2PC based on garbled circuits with GMW in § 4.2.1 and show that the former is more efficient in all settings. Then, we explain our choice of 2PC protocols based on garbled circuits in § 4.2.2 and the applications in § 4.2.3. We describe the parameters that define the amount of data to be transmitted, the properties that impact the communication performance and we list recent optimisations relevant for our improvements on the transport layer, which we integrate into Transputation. As described in § 2.1, there are three adversarial notions that are most commonly used in secure computation: semi-honest, malicious and covert adversaries. This chapter focusses on 2PC secure against semi-honest adversaries.

4.2.1 Yao vs. GMW

Currently the most efficient technique for secure two-party computation is based on Yao's protocol with garbled circuits (see preliminaries in § 2.2 and § 2.3). The main features of Yao's protocol are: it has constant rounds, and it is computationally cheap since it mostly uses lightweight symmetric-key operations and very few expensive public-key operations such as exponentiations. Moreover, due to the nowadays ubiquitous presence of the Advanced Encryption Standard New Instructions (AES-NI) in modern CPUs, the computational overhead of generating and evaluating garbling circuits has further decreased.

The main alternative to Yao's protocol is the GMW protocol [GMW87] (which is purely based on OT). The most efficient GMW protocol known is the one presented by Schneider and Zohner [SZ13] and then refined by Demmler, Schneider and Zohner [DSZ15]. The OTs are precomputed [Bea95] and the base OTs are extended based on the OT extension protocol. The implementation of this work, available as part of ABY library [DSZ15], includes the use of multiplication triples [Bea91], which is generated by an 1-out-of-2 OT in an offline phase [Gil99], to evaluate AND gates.

		Protocols	
	Setting	Yao	GMW
AES	LAN	7	27.88
	EU-US	130.4	2917.3
	EU-AUS	377.92	7325
SHA1	LAN	14.5	1651.27
	EU-US	144.41	187080.6
	EU-AUS	396.52	493143.1

Table 4.1: Yao vs. GMW in LAN and intercontinental settings (runtime in ms).

Unlike Yao’s protocol, the GMW protocol has non-constant round complexity (proportional to the depth of the circuit to be evaluated). This requirement hinders its efficiency, especially in high latency setting. Schneider and Zohner [SZ13] concluded that GMW protocol with multiple rounds was more efficient than Yao’s garbled circuits in low latency LAN settings. However, the evaluations were compared to that of Huang et al. [HEKM11] from 2011. Since then, the garbled circuits protocol has been further optimized. In Table 4.1, we show evidence that Yao’s protocol is currently more efficient than GMW in all network settings. Thus, we do not consider GMW any further in this chapter.

4.2.2 Garbling Schemes

Once we have fixed our attention on Yao’s protocol, we can consider several optimized garbling schemes that have been proposed. In particular, the original garbling scheme by Yao was sub-optimal both from a communication point of view (as it produced large garbled tables for all gates) and from a computational point of view (as the evaluator had to decrypt multiple ciphertexts for each gate in the circuit). Since then, many novel and optimized garbling schemes have been proposed.

We focus on the recently optimized garbling schemes. The three main characteristics of such garbling schemes are: the size of the produced garbled circuits (which impacts the communication efficiency of the protocol), the number of encryptions and decryptions that have to be performed for generating and evaluating a garbled gate (which impacts the computational overhead of the protocol), and the computational assumptions under which the garbling scheme is proven secure.

We have included the three most representative garbling schemes with respect

Protocols	Optimizations
[GLNP15]	Pipelining + Key Scheduling; 4-2 GRR + XOR-1 Pipelining + Key Scheduling; Free-XOR; Half-gates
[ZRE15]	Fixed key AES; Free-XOR; Half-gates

Table 4.2: List of implementations.

to efficiency and security. All other schemes in the literature are strictly worse when it comes to either efficiency or security. Both security and efficiency are crucial properties for a garbling scheme. In our evaluations with Transputation, we extend upon the work of Gueron et al. [GLNP15] to show that with a better selection of transport layer protocols, secure computation with standard assumptions can be made more efficient.

These three implementations exhibit different network characteristics (e.g., the crypto-material that needs to be exchanged). Concretely, we have implemented 2PC protocols based on the following three garbling schemes in Transputation (Table 4.2):

1. The most efficient known garbling scheme that makes conservative and standard assumptions (namely, that AES is a pseudo-random function (PRF)), proposed by Gueron et al. [GLNP15], henceforth GLNP15;
2. A more communication efficient garbling scheme that supports free-XOR [KS08] and half-gates [ZRE15] (at the price of having to assume circular security properties of AES);
3. A further improvement in the computational overhead at the price of having to assume that AES with a fixed-key behaves like an ideal permutation [BHKR13].

We have chosen implementations that range from using minimal and standard cryptographic assumptions to strong and non-standard cryptographic assumptions. Not only do these implementations provide varying degrees of security, but they also differ in terms of the computation performed and the data communicated. The communication depends on the garbled circuit size, which is conditioned on the cryptographic assumption used to generate the garbled circuit. We use the garbled circuits implementations that are part of the Libscapi library [EFL12], since all the known garbled circuit optimizations have been incorporated into it.

GLNP15. The first implementation we consider is GLNP15. The main advantage of this garbling scheme is that it only makes conservative computational assumptions, i.e., it can be proven secure under the assumption that AES behaves like a PRF. Similar to all garbling schemes that we consider in Transputation, the garbling of linear gates (e.g., XOR) and non-linear gates (e.g., AND) is performed differently. In GLNP15, garbling an AND gate produces two ciphertexts (using the *4-2* Garbled Row Reduction technique, or *GRR* for short), while garbling an XOR gate produces one ciphertext using the *XOR-1* technique. From a computational point of view, garbling with AES key scheduling (expansion of the key into separate round keys) is pipelined. Circuit garbling requires four key schedules per gate while circuit evaluation requires two key schedules.

Half-Gate. The second implementation is based on the work of Zahur, Rosulek and Evans [ZRE15], and uses the so called “half-gate” optimization, which in turn is compatible with the “free-XOR” optimization of Kolesnikov and Schneider [KS08]. This optimization requires a stronger computational assumption on AES that assumes some form of circular-security (a kind of related-key assumption). To understand the need for this assumption, we briefly recall the free-XOR optimization. In free-XOR, for each wire i , the keys (k_i^0, k_i^1) are set to $(k_i, k_i \oplus R)$ where R is a global “offset”. To garble a XOR gate with input wires i, j and output wire ℓ , the output key k_ℓ is set to $k_i \oplus k_j$. To evaluate the XOR gate it is not sufficient to compute $k_i^a \oplus k_j^b$ which is equal to $k_\ell^{a \oplus b}$ because of how the keys were set-up. When garbling AND gates, one has to encrypt the output keys under the corresponding input keys. Lets assume, without loss of generality, that the evaluator has input keys k_i^0, k_j^0 for this gate and therefore learns k_ℓ^0 . Now the evaluator should not be able to decrypt the value k_ℓ^1 which is encrypted under keys k_i^1, k_j^1 , e.g.,

$$E(k_i^1, k_j^1; k_\ell^1) = E(k_i^0 \oplus R, k_j^0 \oplus R; k_\ell^0 \oplus R) .$$

Since all the 0-keys are known, the only unknown left is the global offset R which is, essentially, encrypted under R itself; thus, explaining the need for the circular security assumption. The half-gate optimization reduces the number of ciphertexts necessary to garble an AND gate from four to two, using a different approach than GLNP15. We refer to the original paper [ZRE15] for details.

JustGarble. The final implementation we consider was proposed in the JustGarble framework [BHKR13]. Recall that key-scheduling is the most expensive phase when using the AES-NI, i.e., the instruction is optimized to garble large amount of data under the same key, but loses some of its efficiency when different keys have to be used every time. In garbling schemes, each gate consists of ciphertexts where different keys are used; thus, the full power of the AES-NI set is not exploited. In JustGarble, AES is used as an ideal permutation “in stream cipher mode” by setting the key as a fixed constant. To encrypt a message m under a key k , we compute $C = AES_c(k) \oplus m$ for some constant c . For this construction to be secure, we need to assume that AES with a fixed-key behaves like a random permutation (this in turn implies circular security as needed for free-XOR and half-gates). Fixed-key AES requires one key schedule to be computed for the entire circuit and eliminates the need for multiple AES key scheduling, which is relatively more expensive than encryption and decryption. This usage of AES is non-standard and, amongst the three schemes presented, it provides the most extreme efficiency/security trade-off.

4.2.3 Applications and Circuit Size

Once we have fixed the protocol and the garbling scheme, we are left with one dimension, namely, which function should we evaluate using the 2PC protocol. For garbled circuit protocols, the circuit size plays a significant role in defining the amount of data to be transferred over the network. The amount of data transferred from the Garbler to the Evaluator is a linear function of the circuit size. In particular, there is a difference in the price to pay (in terms of communication complexity) for linear gates vs. non-linear gates, and different garbling schemes have different coefficients for these two types of gates.

In this chapter, we consider applications with circuits of three different sizes. These circuits are becoming the de-facto standards for benchmarking of secure computation protocols, mostly since they represent three different orders of magnitude in circuit sizes. In particular, we benchmark Transputation on the circuits for AES ($\approx 10^5$ gates), SHA256 ($\approx 10^6$ gates) and MinCut ($\approx 10^7$ gates), which are considered as applications of small, medium and large circuits respectively. AES has been used as a benchmark for secure computation protocols [GLNP15] and MinCut is a large circuit that allows us to show that the performance of the protocols in different network settings varies depending on the circuit size. We use MinCut

	Gates	
	AND	XOR
AES	6,800	25,124
SHA256	90,825	42,029
MinCut	999,960	2,524,920

Table 4.3: Number of boolean gates per function.

	Function		
	AES	SHA256	MinCut
GLNP15	0.59	3.41	69.04
Half-Gate	0.21	2.77	30.52
JustGarble	0.21	2.77	30.52

Table 4.4: Circuit size per function in megabytes for each garbling scheme.

circuit as it is the largest circuit available in libscapi. We use SHA256 circuit as an intermediate-size circuit. The exact number of gates and the distribution between AND and XOR gates for these circuits is shown in Table 4.3.

For a particular circuit, the circuit size depends on the security assumptions. Depending on the security assumptions, the amount of data for a particular circuit varies. When garbling using GLNP15, each AND gate produces two ciphertexts while each XOR gate produces one ciphertext. For AES, SHA256 and MinCut, the amount of data communicated is 0.59MB, 3.41MB, 69.04MB respectively. When garbling using the Half-Gate construction or JustGarble, no ciphertexts are needed for XOR gates while two ciphertexts are produced per AND gate. For AES, SHA256 and MinCut, the amount of data communicated is 0.21MB, 2.77MB and 30.52MB. It can be observed that, for MinCut circuit with many XOR gates, the number of ciphertexts sent over the network has a high dependency on the garbling scheme used. In Table 4.4, a summary of the circuit sizes in megabytes is provided.

4.3 Transport Layer

In this section, we describe in § 4.3.1 the network characteristics that Transputation measures to optimise performance, parameters for selection of optimal transport

layer protocol, and the challenges of integrating transport layer protocols. Then, we describe in § 4.3.2 the transport layer protocols that we integrate into Transputation.

4.3.1 Transport Protocol Selection

Given the circuit size and the 2PC implementation, the transport layer of Transputation measures the packet loss and the latency to heuristically determine which transport protocol is optimal. The decision whether reliability and congestion control mechanisms are needed is made considering the network characteristics. Transputation runs continuous experiments with PING, probes the network for losses, tries different sending rates and selects a protocol that empirically produces optimal performance.

Protocol selection based on latency. Latency plays an important role in the choice of transport layer protocol. When the time to transmit one TCP window is longer than the RTT, the transmission proceeds in full pipe, and is essentially similar to UDP since the congestion window does not limit the transmission. To determine if transmission proceeds in full pipe, Transputation performs the following computation: let W be the bytes in the TCP window and let t_{trans} be the transmission delay of one byte. Let RTT be the time it takes to transmit one TCP segment and receive an ACK. If $W \cdot t_{trans} > RTT$, then there is no impact of TCP congestion window on the latency since transmission proceeds in pipeline. Transputation measures the RTT, the window size W and the ratio of $W \cdot t_{trans}$ to RTT and determines which transport protocol to use (i.e., window-based or to transmit in full pipe).

Protocol selection based on communication rounds. The relevant parameters here are the number of rounds and data volume. Window based protocols, such as TCP, are not optimal for 2PC implementations with small number of rounds and large data volumes due to the fact that the transmission window of TCP increases with the number round trips. In TCP the window starts with one segment and increases exponentially with every received ACK. As a result, although only a few interactions are required on the application layer, they will be performed in multiple interaction rounds on the transport layer. This factor is most evident in high latency networks.

Avoiding Packet Loss. Transputation avoids packet loss at the sender by adjusting the size of buffers as a function of latency, transmission rate and the data volume to be transmitted. The application sends packets to the transport layer, which depletes the buffer by passing the packets on to the IP layer to subsequently transmit the packets on the wire. When the application passes the packets faster than the transport and the IP layers can process them, then the buffers will overflow and packets will be lost. Transputation adjusts the buffers, according to the computation below, to avoid packet loss. Given the differences between the transmission rate and the rate at which the data is passed on to the IP buffer (respectively transport layer) at the sender, the transmission rate and the rate at which the IP buffers (and respectively transport layer) are depleted at the receiver, and the input sizes and the secure computation implementation (which define the data volume and the rate at which it will be exchanged), Transputation performs a computation of the maximum amount of data that can be sent in one window.

The computation that is performed by Transputation is as follows: Given a receiver buffer of size B , with data arrival rate $R_{arrival}$, and the buffer depletion rate R_{read} . Transputation computes the maximum window size as geometric series that converges to: $L = \frac{B}{1 - R_{arrival}/R_{read}}$. During the computation, the data transmitted will be limited by L bytes during each transmission window. This accounts for the data that is being read, while new data arrives, and allows to optimise the communication. This is not the same as the flow control performed by TCP, which avoids overflow at the receiver.

4.3.2 Transport Protocols in Transputation

We integrate the following transport protocols into Transputation: TCP, UDP and SABUL. We implement TCP for comparison to other transport protocols. We use UDP as a benchmark for connection-less protocols on networks without losses. We integrate SABUL as it is currently used by an increasing number of applications, and provides reasonable performance for Internet communication. Furthermore, SABUL is TCP-friendly and fair to other applications sharing the same network. New and even more efficient protocols may be developed in the future. Transputation enables easy integration of new and additional transport protocols. We describe the steps needed for integrating new protocols into Transputation in § 4.4.

4.4 Transputation Implementation

The design goal of Transputation is to provide a modular design that can be extended with other secure computation protocols as well as transport layer protocols. Transputation allows secure computation researchers to focus on the protocol details without worrying about the networking aspects of the implementation. Modularity is not restricted to secure computation protocols. Transport layer protocols, ancillary to those included in the framework, can be added to the framework if required.

Abstraction. The transport layer part of Transputation is implemented as a wrapper written in C++ that can be easily plugged in with secure computation protocols. The current version uses synchronous sockets that is sufficient for our purposes. It abstracts the network functionality and removes the requirement to deal with the transport layer protocols themselves. The transport layer protocol can be set at runtime, making it easier to compare different protocols without the need for recompilation. Since most of the secure computation implementations developed in the past few years are written in C++, polymorphism in C++ is used to achieve modularity. We implement an abstract class with methods required to establish and close connections, and to send and receive data. Every transport layer protocol that is or will be implemented in our wrapper extends this class and implements these methods. This provides secure computation developers with two benefits: First, they do not need to know how to use the transport layer protocol and second, they can use new protocols that are added to the wrapper without making any change to the executable or library of the secure computation implementation. To implement a new protocol only the four methods of the abstract class are required: `SetupClient`, `SetupServer`, `RecvRaw`, `SendRaw`.

The wrapper currently supports UDP, TCP and SABUL. Since UDP does not provide reliability, it cannot be used in real-world scenarios with packet losses or where the correct order of packets cannot be guaranteed. If a dedicated network without packet loss is available, then UDP can be used. In all other cases, we recommend to use TCP or SABUL in Transputation. To choose between TCP and SABUL, we provide evaluations in § 4.5.

Simplification. We have used predefined class methods to simplify common tasks. When an instance of the `Transport` class is created, a socket is already allocated and set up on creation (by using `socket()` for TCP or UDP). Then the user decides if a client that connects to a server should be created, or if a server

```
1      auto *t = GetTransport("tcp");  
2      t->SetupServer("0.0.0.0", 1234);
```

Figure 4.3: Setting up a server with our wrapper.

that listens on a port and waits for incoming connections should be created. For example, using our wrapper, a TCP server can be setup as shown in Figure 4.3. This reduces the number of lines needed as well as improves the readability and encourages users to separate program logic from network code. This is important to make the code reusable. It is also easier to test the functionality of different parts when they are separated in a modular design. The wrapper also includes two static methods, `GetLatencyClient()` and `GetLatencyServer()` to measure the latency. These methods use UDP packets to measure the RTT in milliseconds. This can be used to decide which protocol should be used. Finally, the wrapper takes care of byte order, which makes it simple to port applications to different platforms.

Packet handling. Transport layer protocols have contrasting methods to send data. For instance, UDP sends single packets, and hence, sending ten 100 byte packets is not an issue. However, sending packets larger than the maximum allowed packet size, limited by the underlying maximum transmission unit of the network stack, is not possible without splitting the data into smaller chunks. This has to be done by the program that incorporates UDP, which does not provide this by itself. In contrast to UDP, TCP sends data as a stream. If the data to be sent is too large, it will be split into multiple packets by the protocol. To simplify usage of different transport layer protocols, the wrapper allows users to send and receive packets in arbitrary sizes.

To solve the issue where multiple packets are received as a big chunk, the wrapper includes a `Packet` class that can be used to send a given amount of data. The receiver can, without knowing the packet size prior to receiving, receive the packet. For all protocols, which can be included in the wrapper, the data will be split into multiple packets if needed and reassembled at the receiver. When 10 packets are sent with the wrapper it will receive 10 packets. In order to tell the packets apart, the length information is added to the data so it can be interpreted by the receiving side. This length information is converted to network byte order to make it cross compatible among systems with a different endianness. So sending data from a big-endian machine to a little-endian machine (or vice versa) will work since the

```
1      auto *t = GetTransport("tcp");
2      t->SetupClient("0.0.0.0", 1234);
3      t->Connect();
4      auto p = t->Recv();
5      t->Send(p);
```

Figure 4.4: Example of an echo client using our wrapper.

bytes of the 16 bit length value are not accidentally flipped due to different byte order.

Integration. The wrapper can be integrated into any secure computation protocol by simply calling methods of the wrapper such as `Send()` or `Recv()` in the `Transport` instance. The protocols can be supplied as a string such as `udp` and `tcp`. If the wrapper is extended with a new transport layer protocol, then the framework will be able to use this protocol without any changes. For example, an echo client that connects to a server, receives a packet and echos it can be implemented with the code in Figure 4.4. We have integrated the wrapper into libscapi. Libscapi uses external OT extension libraries that has its own network code. By incorporating our wrapper in libscapi, both—libscapi code and OT extension code—can use suitable transport layer protocols without changing the rest of the code. This is an advantage of using a common network wrapper. Transport layer protocols can be added to the wrapper and used by both. With our approach, we want to separate the network code from the application code.

4.5 Simulations and Evaluations

In this section, we provide the simulation results which are used to understand the effect of latency, packet loss and bandwidth on the performance of secure computation protocols (§ 4.5.1). Then, we describe realistic deployment scenarios (§ 4.5.2) followed by evaluation results obtained in LAN and WAN settings (§ 4.5.3).

4.5.1 Simulations

We simulate two parties performing secure computation on a single machine. We simulate latency and packet loss using `tc qdisc` network emulator on a single virtual

machine (VM) instance from Vultr ² with a 64-bit single core CPU with 2.6 GHz and 2 GB RAM. We use the loopback interface for communication. The simulation results provide the benchmark for the executions in real network setups that we describe in § 4.5.3.

Past works have compared performances between LAN and WAN settings. For practical deployment, we need to understand the network conditions in finer granularity. Through these simulations we aim to understand the impact of latency and packet loss on 2PC protocols. We simulate latencies between 1ms–300ms, packet losses between 0.01%–0.05% and bandwidth of 200Mbps, 500Mbps, 1Gbps and 10Gbps. The latencies were chosen to represent communication between machines on the same network as well as those in different parts of the world. For instance, the RTT within North America is on average 50ms, RTT between machines on either side of the Atlantic Ocean is about 100ms and machines placed in North-America and Asia or EU and Australia is about 300ms. Packet losses were chosen such that they are representative of realistic packet losses observed in networks ³. Bandwidths were chosen based on measurements performed using `iperf` on different networks—local and across continents.

To understand the effect of latency on 2PC protocols, we run them using TCP, UDP and SABUL. As reliable communication is required to satisfy the correctness property of secure computation in real networks, we use UDP to benchmark the runtimes achievable when the bandwidth is optimally utilized. We use two reliable transport layer protocols, TCP and SABUL, to show the bandwidth utilisation for different circuit sizes. All experiments use single thread with AES-NI and the results are the average of 100 runs.

For a small circuit such as AES, it can be observed in Figure 4.5 that TCP with Nagle’s algorithm [Nag84] disabled performs better than SABUL. For small circuits, few kilobytes of data are sent over the network while SABUL is optimized for large data transfer. For a medium-sized circuit such as SHA256, it can be observed in Figure 4.5 that TCP performs better than SABUL using JustGarble and half-gate protocols, while with GLNP15 protocol, the performance of SABUL is better than TCP as RTT increases. Our observation is due to a combination of reasons: the performance of SABUL is better as bandwidth-delay product increases as well as

²<https://www.vultr.com/>

³<http://www.verizonenterprise.com/about/network/latency/>

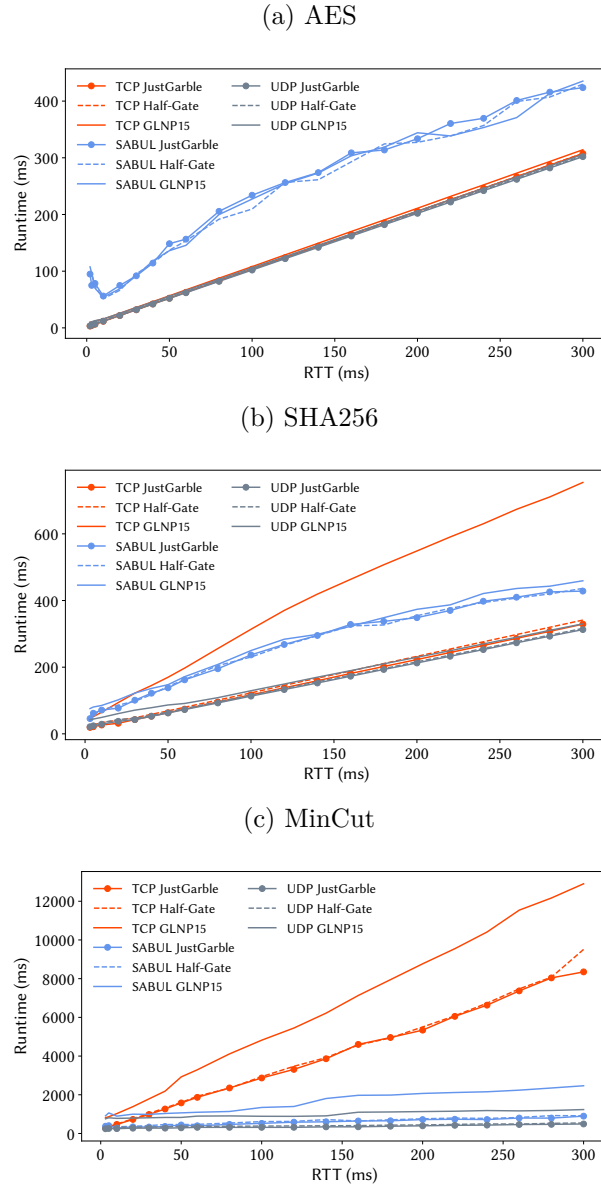
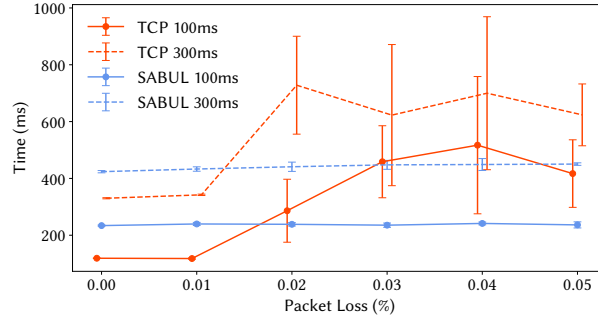
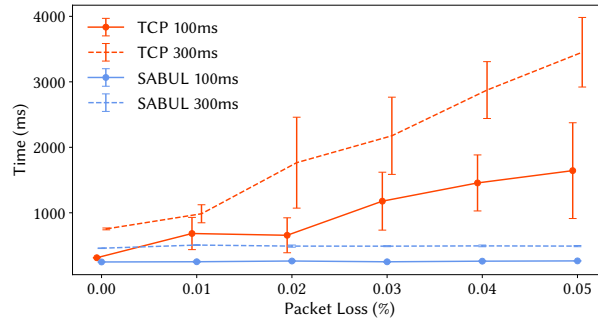


Figure 4.5: Effect of latency on a 10Gbps link for (a) AES (b) SHA256 (c) MinCut

(a) AES



(b) SHA256



(c) MinCut

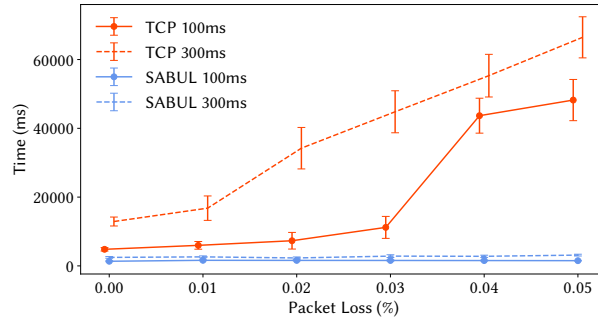


Figure 4.6: Effect of loss on a 10Gbps link for (a) AES (b) SHA256 (c) MinCut

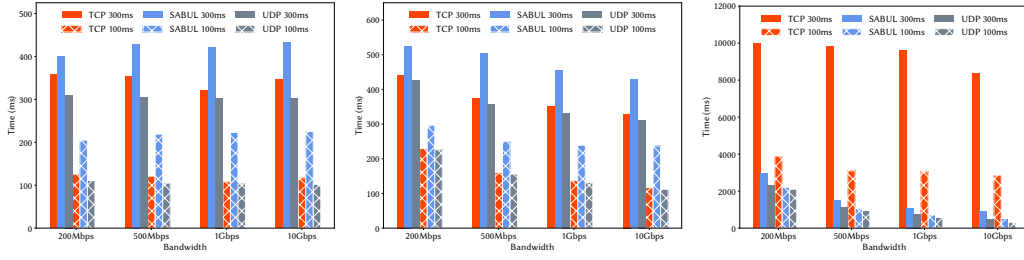


Figure 4.7: Performance of JustGarble protocol at various bandwidths for (a) AES (b) SHA256 (c) MinCut.

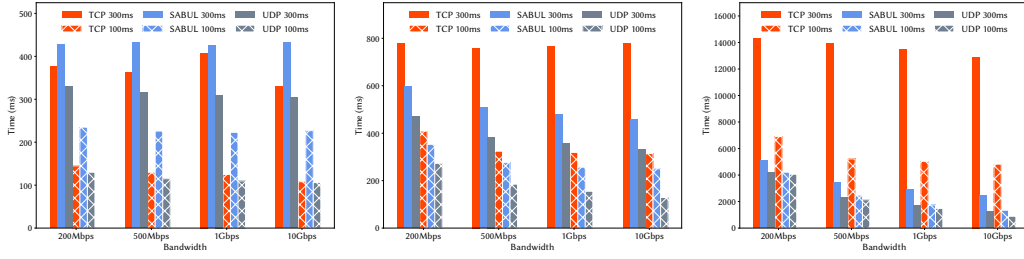


Figure 4.8: Performance of GLNP15 protocol at various bandwidths for (a) AES (b) SHA256 (c) MinCut.

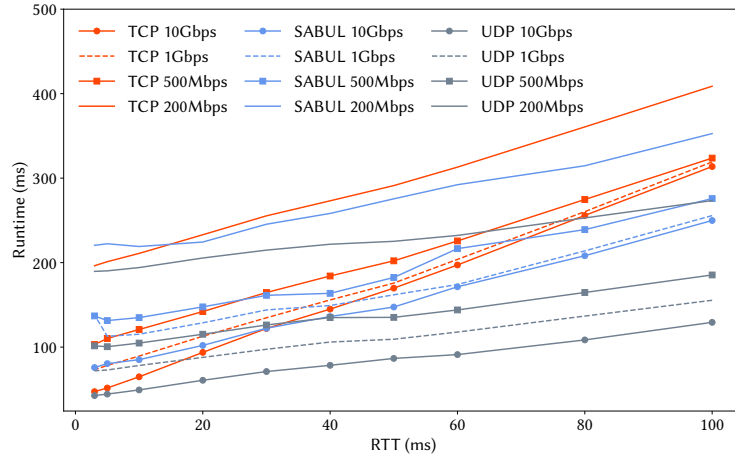


Figure 4.9: Performance of SHA256-GLNP15 in different bandwidths and latencies.

when the amount of data transferred increases. For a large circuit such as MinCut, the performance of SABUL and TCP is quite different from that observed for AES and SHA256. It can be observed in Figure 4.5 that SABUL utilizes the available bandwidth much better than TCP. As many packets are sent from the Garbler to the Evaluator, the congestion control mechanism plays an important role in controlling the rate of packet transmission. The increase in latency affects the performance of TCP more than SABUL as SABUL uses a timer-based selective ACK instead of reacting to packet level events.

When considering packet loss, for small circuits such as AES, loss rate that we consider impacts the performance of TCP more than SABUL, as can be seen in Figure 4.6. For medium-size circuits such as SHA256 and large circuits such as MinCut, increase in loss rate deteriorates the performance significantly when TCP is used. On the other hand, SABUL handles packet loss better and the performance deterioration is very low. For medium and large-sized circuits on a network with at least 100ms delay, SABUL is more suitable than TCP. We also plot the standard deviation of the runs as the variance is non-negligible in many of the results.

We also present the effect of bandwidth on the three applications we have considered. Figure 4.7 is the results for JustGarble protocol while Figure 4.8 is the results for GLNP15 protocol. As the transport protocol providing better performance for SHA256 using JustGarble and GLNP15 protocol differs at high latency, finer measurements can give us insight on the latency at which one transport protocol performs better than the other at different bandwidths for GLNP15 protocol. In Figure 4.9, we observe that SABUL performs better than TCP even at a latency of 40ms when the available bandwidth is 10Gbps. As the available bandwidth decreases, SABUL performs better at lower latencies. For instance, at 200Mbps, SABUL is better than TCP at 20ms.

4.5.2 Deployment Setups

For performance evaluation, we setup deployments in LAN setting and WAN setting. These settings provide two common setups of evaluation of secure computation protocols. In both setups, we use two VM instances from Azure running Ubuntu 16.04, each with a 64-bit Intel Xeon quadcore CPU with 2.4 GHz and 28 GB RAM. **LAN setting.** We ran the experiments on two machines located in the same data centre in the EU using high-bandwidth network and low latency. The latency was

0.5 ms on a 10Gbps link. The variance was within 10%.

WAN setting. We ran experiments on two pairs of locations with different latencies. These locations were chosen to show the behaviour of secure computation protocols with different transport layer protocols as latency increases.

EU-US: In this setting, one machine was located in the EU while the other was located in central US. The latency was 110ms and the network speed was estimated to be 1Gbps. The measured speed for a single TCP connection was 200Mbps on average. The variance was within 15%.

EU-AUS: In this setting, one machine was located in the EU while the other was located in south-east Australia. The latency was 300ms and the network was estimated to be 1Gbps. The measured speed for a single TCP connection was 100Mbps on average. Variance of 20% was observed in this setting for secure computation of AES and SHA256 while the variance for MinCut was 30% for TCP and 25% for SABUL.

4.5.3 Experimental Evaluations

In this section we present the results obtained by using Transputation for the three secure computation protocols using TCP and SABUL for communication in LAN and WAN settings. All experiments were run using single thread and AES-NI. The results in the following section are the average of 100 runs.

We summarize the experimental results in Table 4.5. The runtimes in the table include the garbling time, transfer of data from the Garbler to the Evaluator and the computation of output. In LAN setting, TCP performs best for all circuit sizes. When Nagle’s algorithm [Nag84] is disabled, the packets are sent as soon as they arrive at the buffer. Disabling Nagle’s algorithm is advantageous in LAN setting as the communication is fast and computation accounts for the bulk of the runtime.

In the WAN setting, when 2PC protocols are run between Azure VMs in EU and US, the circuit size begins to influence the performance. For AES and SHA256, TCP is still more efficient than SABUL but the tide tilts for MinCut. MinCut using SABUL is $2.7 - 3\times$ faster than TCP. In fact, secure computation of MinCut with GLNP15 protocol using SABUL is more efficient than using JustGarble or half-gate protocol using TCP.

In the WAN setting, when the 2PC protocol is run between Azure VMs in EU and Australia, the influence of latency on 2PC of large circuits becomes much

	Setting	Just garble		Half-gate		GLNP15	
		TCP	SABUL	TCP	SABUL	TCP	SABUL
AES	LAN	2.4	187.1	2.9	191.3	7	202.7
	EU-US	127.4	403.9	126.3	408.3	130.4	444.2
	EU-AUS	312.44	566.84	310.88	580.2	377.92	592.5
SHA256	LAN	13.5	191.9	19.9	226.7	30.5	233.45
	EU-US	146.23	332.24	151.99	318.26	266.46	411.96
	EU-AUS	362.53	568.22	394.13	587.03	650.44	612.43
MinCut	LAN	255.19	598.2	267.2	740.2	700.8	1255.9
	EU-US	2616.59	896.74	2783.6	957.6	4911.89	1802.25
	EU-AUS	8204.61	1068.07	8693.57	1163.14	13805.2	2001.27

Table 4.5: Experimental results for garbled circuits protocols (runtime in ms)

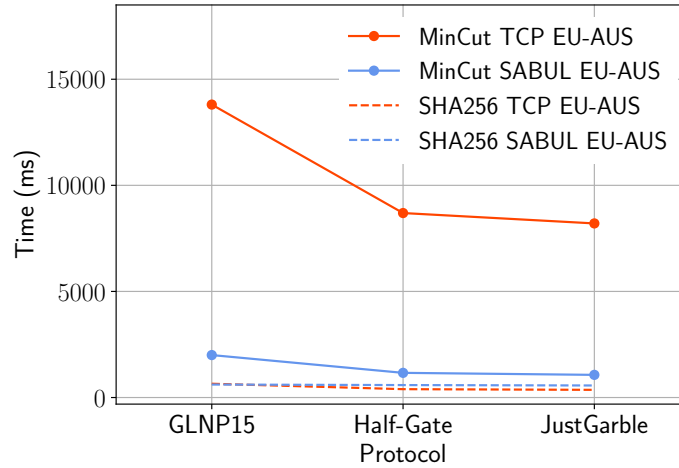


Figure 4.10: Comparison of assumptions.

more evident. Secure computation of SHA256 with GLNP15 protocol (with 223,679 ciphertexts) using SABUL is only a little faster than using TCP. Secure computation of MinCut using SABUL is 7 – 8 \times faster than using TCP. This is significantly more than the improvements that can be expected from secure computation protocol improvements [ZRE15]. In Figure 4.10, we provide a comparison of the performance of TCP and SABUL for the three 2PC protocols that further emphasizes the improvement in performance by using appropriate transport layer protocols.

The comparison of TCP and SABUL with respect to increasing ciphertext size can be seen in Figure 4.1. The impact of latency on the two transport layer protocols becomes apparent from this figure. When using SABUL, time taken by 2PC protocols increases by 200ms when latency between the two parties increases by 200ms. The performance of TCP deteriorates much more significantly as latency increases.

4.6 Related Works

In Figure 4.11, we provide a graphical illustration of related work on 2PC and put our work in perspective. The landscape of secure computation protocols is broad and diverse, and different protocols exist for different number of parties, adversarial models, corruption thresholds, etc. In this chapter, we have chosen to focus on the most natural case of secure two-party computation.

Garbled Circuits Optimizations. Several works have improved the efficiency of garbled circuits. The original protocol by Yao [Yao86] requires a transfer of four ciphertexts per Boolean gate in the circuit, and requires four decryptions for the evaluation of a garbled gate. The number of decryptions per gate in the evaluation phase was reduced to one due to the *point-and-permute* strategy [BMR90], which also reduces the size of the ciphertexts by a factor of two (approximately). The main measure of *communication complexity* of garbling schemes is the number of ciphertexts which are transmitted per Boolean gate. This was first reduced by Naor, Pinkas and Sumner [NPS99] to three ciphertexts per gate using the so called 4-3 GRR technique, and to two ciphertexts per Boolean gate by Pinkas et al. [PSSW09] (4-2 GRR). Thanks to the *free-XOR* technique [KS08], it is not necessary to transfer any ciphertexts for XOR (or other linear) gates. Unfortunately, the *free-XOR* technique was incompatible with the more advanced 4-2 GRR technique. The two techniques were finally combined thanks to the *half-gate* optimization, which combines the benefit of *free-XOR* (no ciphertexts for XOR gates) with the advanced 4-2 GRR technique (only two ciphertexts per AND gate). Unfortunately, the *free-XOR* technique is only secure under non-standard cryptographic assumption [CKKZ12] and, in particular, it requires some form of “circular security” assumption. Using an even stronger assumption, i.e., *fixed-key* AES behaves like a *random permutation*, faster garbling schemes were proposed by Bellare et al. [BHKR13].

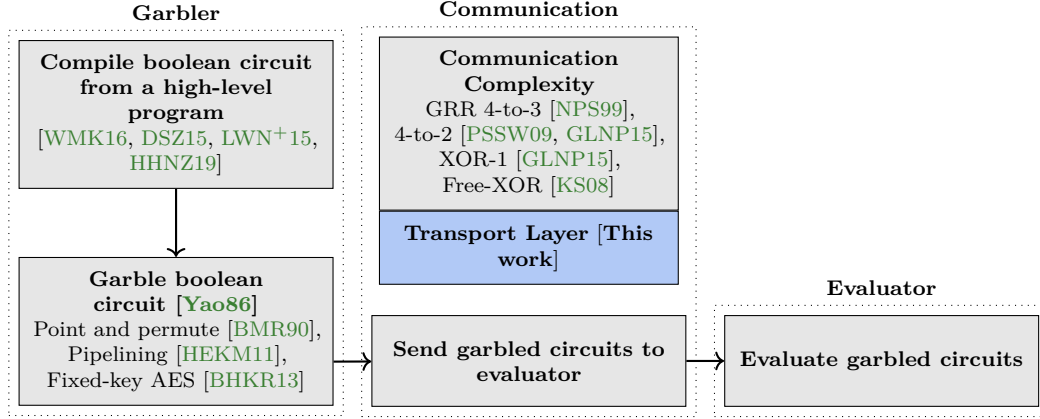


Figure 4.11: Positioning our work within related work.

While efficiency is a crucial aspect in 2PC, some have questioned whether it is wise to make protocols efficient at the cost of strong assumptions. In particular, we note that a conservative approach is usually adopted by the industry as it is difficult to change protocols if vulnerabilities are discovered after deployment. Therefore, Gueron et al. [GLNP15] provided novel constructions of garbled circuits that can be proven secure using standard assumptions and that require two ciphertexts for AND gates (4-2 GRR) and one ciphertext for XOR gates (XOR-1). It concludes that the price to pay for the stronger security guarantees in practice is much less than it is in theory. Our evaluations in some sense confirm and strengthen these conclusions: our experiments show that the choice of the right transport protocol has a much greater impact on overall efficiency than gambling on security by using non-standard assumption. In particular (when evaluating large circuits over WAN), using SABUL plus standard assumptions is $4\times$ faster than using TCP plus non-standard assumptions.

Nielson, Schneider and Trifiletti [NST17] (in the active security setting) attempts to improve the communication performance by saturating the network through the use of multiple parallel TCP connections. Though multiple TCP connections improves the performance, arbitrary number of parallel connections could lead to network congestion and lower throughput [HAN02].

Secure Computation Frameworks. Since Fairplay [MNPS04] was implementation in 2004, various frameworks have been developed for secure 2PC [HEKM11,

[EFLL12](#), [DSZ15](#)]. Currently, a garbled circuit framework with security against passive and active adversaries is provided in Libscapi [[EFLL12](#)] library with the latest optimizations. Though this chapter only considers Boolean circuits, we note that a framework for mixed protocol was implemented in ABY [[DSZ15](#)]: this framework combines arithmetic sharing, boolean sharing and garbled circuits, and allows for conversions between the sharing methods. All previous frameworks focus on secure computation and use standard TCP socket provided by the operating system.

The recent work of Hastings et al. [[HHNZ19](#)] surveys general purpose compilers for secure computation, which provide high-level abstractions to describe functions in an intermediate representation (such as a circuit). While it focuses on compilers, we focus on efficiency of protocol execution. Hence, the two works address orthogonal problems. The sample programs it chooses do not represent practical MPC use case and instead attempt to test the usability of the compiler code and how easy it is to write example code for an application for the compiler. Furthermore, the tests are performed on standalone environment and do not account for the context, such as the network condition, in which the protocol is run. The goal of that paper is not to provide a practical testing framework, neither is it to test the efficiency of the protocols. Instead, its goal is to make explicit which compilers are written for experts and which for non-experts.

The recent work by Barak et al. [[BHKL18](#)] considers the possibility of providing MPC as a service where users use the platform to run protocols. They provide an environment where users can participate in a low-bandwidth MPC protocol using web-browser or an app on the phone. While they focus on low-bandwidth MPC protocols, which require multiple rounds, and present evaluations only in LAN setting, we focus on high-bandwidth constant round 2PC protocols (garbled circuits) in LAN and WAN settings.

Related work shows that significant progress has been made in computation complexity of 2PC implementations. When evaluated on a single host the implementations produce good performance. However, on real networks with other traffic and concurrent processes, latency and packet loss, the efficiency collapses and implementations incur prohibitive latency. In particular, we note that no previous work has considered how to improve secure computation by addressing the issue of transport layer performances, as addressed in this chapter.

Securing DNSSEC Keys via Threshold ECDSA From Generic MPC

The domain name system, standardized in RFC 1034–1035 [Moc87a, Moc87b], maps domain names to IP addresses and provides a platform for an increasing number of systems and applications. Unfortunately, security is not part of its design. It is vulnerable to DNS cache poisoning, where an attacker can introduce incorrect DNS data to direct end users to a different IP address (often that of the attacker) than that of the queried domain name [Bel95, AA04, Kam08, SS10, HS13b].

DNS security extensions, standardized in RFC 4033–RFC 4035 [AAL⁺05a, AAL⁺05c, AAL⁺05b], mitigate cache poisoning using cryptographic techniques. DNSSEC provides *origin authentication* and *integrity* of DNS data, i.e., the machine that sends a DNS query can verify if the response to DNS queries has originated from the intended DNS server and has not been altered in transit. A DNS record is certified using a digital signature scheme, with RSA and ECDSA being the standardized algorithms in RFC 5702 [Jan09] and RFC 6605 [HW12] respectively. While RSA is the most commonly used scheme, the need to prevent fragmentation of UDP responses requires the signing keys to be short. ECDSA is recommended in RFC 8624 [WS19] as it provides the same level of security as RSA but with much smaller signatures.

DNSSEC prevents cache poisoning as long as the operator can be trusted. In practice, very few domain owners run their own authoritative name servers and manage their zones, and this role is usually outsourced to the DNS operators. Outsourcing management of zones comes with several benefits, such as increased

availability and a lower chance of misconfiguration (presumably, because the operators have specialists at hand that administer their name servers). However, when DNSSEC is used, this outsourcing also introduces several issues related to key management. With the introduction of DNSSEC, the operator handles signing keys of domain owners. This not only means that each domain owner needs to relinquish control of their signing keys, but it also makes the operator a lucrative target for attackers. Access to the signing keys gives the operator full access to the zone, i.e., the operator can potentially change the zone file at will and include malicious information.

Centralization of key management. DNSSEC burdens DNS operators with the additional task of generating and managing the keys of their users. Recent work has demonstrated that a large number of domains share the same key [CvRC⁺17]. Sharing the same key across multiple domains makes the DNS provider a lucrative target. If the key of one domain is compromised, several *other* domains can be compromised as well. Another study [SW17] has shown issues with key generation that result in keys with inadequate security.

Centralization of operation. A second issue that arises from the problem of the DNS operator being in-charge of key management is that the entire operation is centralized. In other words, any guarantee towards integrity of a DNS response to a query is lost if the operator is corrupted. This implies that DNSSEC does not prevent attacks from powerful adversaries on the operator, such as nation-state actors. It is possible that compromised keys can result in a complete takeover, for instance, of a TLD [RABP17]. E.g., access to the private part of the ZSK allows an adversary to change the DS record of a child zone and validly sign them. Such an access gives an adversary the capability to make changes that will result in a failed verification of the chain of trust [AM07]. In recent years, several examples of sophisticated attacks on DNS registrars have been observed in Germany [Kre19], Greece [Cim19] and Sweden [Net19] as part of attacks on DNS infrastructure [Tal19].

Threshold Signatures for DNSSEC

Threshold signatures are a natural candidate to solve the issues outlined above. A threshold signature scheme distributes a signing key to n signers such that any subset of at least t signers can sign a message. Since the signing key is distributed

among multiple signers, it will remain private as long as at least t servers remain uncompromised. Moreover, the threshold signing scheme can be made secure against tampering to prevent a malicious operator from compromising a response.

While threshold RSA has previously been studied for fault tolerance in DNSSEC, threshold ECDSA has not been used for DNSSEC in spite of an increased interest in threshold ECDSA in recent years [Lin17, LNR18, GG18, DKLS18, DKLS19]. All of these recent works motivate the problem of threshold ECDSA in the context of crypto-currencies, a problem that is substantially different from DNSSEC: First, recent works on threshold ECDSA focus on full threshold, i.e., privacy of the signing key is maintained when up to $t = n - 1$ signers collaborate. Second, the focus has typically been on malicious security, i.e., signers are not assumed to behave according to the signing protocol. However, it is possible to design faster protocols by relaxing some of these security guarantees by requiring an honest majority, or assuming that signers do not deviate from the signing protocol. The diverse context in which DNS is used can benefit from solutions that are not limited to a specific threat model. Furthermore, there are only a handful of DNS operators that are typically used per zone in the world of DNS, while the identity and the number of parties involved in cryptocurrencies can vary widely. Threshold signatures are more efficient for small number of parties, which allows us to develop a practical solution to a pressing security problem.

In the real world application of DNSSEC, where multiple operators (usually, two or three) serve a domain, the possibility of only one of them being controlled by an adversary is reasonable as operators are often corporations located in different parts of the world and adhere to local laws. In such a setting, a full threshold protocol may not be necessary, and a protocol that assumes an honest-majority among the operators is sufficient. For instance, for a domain with three operators, we assume that none of the operators collude. Moreover, DNS operators are bound by legal contracts with their customers and they provide service according to this contract. These legal bounds allow us to consider operators that do not act maliciously because such an action would be a breach of contract. However, in such a case we still need to protect keys stored at the operator as employees can snoop on them.

Contributions

A summary of our contributions:

- We present a transformation for generic MPC protocols over a field \mathbb{Z}_p to protocols over an elliptic curve group of order p , as is required for ECDSA. This transformation only introduces a small overhead in terms of efficiency and preserves the security properties of the original MPC protocol. We obtain several different instantiations of threshold ECDSA for different corruption models (honest or dishonest majority and passive or active adversaries) that can be used to compute a large number of signatures at a very low cost. This transformation can have other useful applications beyond ECDSA signing.
- The generality of our approach results in efficient key generation for ECDSA, which has been expensive in most of the prior works. This is beneficial in our setting as operators that manage many zones will generate many keys.
- We implement this transformation in MP-SPDZ ¹ [Dat19] to support threshold signing with ECDSA in many different threat models. We benchmark each instantiation against state-of-the-art protocols for threshold signing and show that they either perform as efficiently or better than customized threshold ECDSA protocols.
- We perform the first measurement study to understand the extent to which multiple operators are used in the Internet. We find that 40% of the domains in the Alexa Top-100 use multiple operators while the proportion of domains in the top-million is 3.5%. Our measurements show that there are thousands of domains that use multiple operators where our threshold ECDSA protocols can be used.
- We describe and implement a full system, by integrating MP-SPDZ with Knot, which can be used by multiple DNS operators for the purpose of implementing DNSSEC, such that no operator has access to the signing key.

¹SPDZ pronounced as ‘speedz’.

Chapter Outline

Section 5.1 presents a measurement study we performed. We show that a significant number of domains use multiple operators, which allows them to use our solution. Section 5.2 outlines our system and threat model. Section 5.3 presents our technical contribution towards designing efficient threshold signing protocol. Section 5.4 shows how we integrate our signing protocol into a well-known DNSSEC application. Section 5.5 presents a number of experiments and compares the results with prior work. Section 5.6 discusses how our work relates to prior works.

5.1 Quantifying Multiple Operators

Since the large distributed denial of service (DDoS) attacks on Dyn [DYN16] and NS1 [Bee16] in 2016 that overwhelmed these managed DNS operators, many domains are using multiple operators to increase the redundancy of the zone so that they do not fall victim to another DDoS attack. However, no recent work has measured the number of domains that make use of multiple operators. As we propose to use our multiparty ECDSA protocol for DNSSEC zone signing, we measure the extent to which multiple operators are used on the Internet. We consider a domain to have more than one operator if the DNS name servers of the same domain are hosted by an entirely different DNS operator.

5.1.1 Data Collection Methodology

If a domain name is configured to be served by three DNS name servers, we check whether it is managed by the same operator. For our purpose, we are interested in nameservers run by different operators and not necessarily name servers placed at different locations. For instance, some domains might use two operators who are geographically located in close proximity to each other; sometimes, even in the same data centre. We are interested in the setting with different operators as they do not have a business relationship with each other that will allow them to pass on copies of signing keys. Hence, being geographically close does not eliminate the need to run a secure signing protocol. Some domains use a single operator that has name servers at different locations. In principle, a multiparty ECDSA protocols can be used in this setting as well because it provides better security than simply storing a copy of the signing key on each name server.

Our measurements were conducted using the Alexa Global Top-1m (top 1 million websites) list as the dataset [Ale19]. The list was downloaded on 12 July 2019. We ran scans on the same date on all the domains in the dataset and requested its NS records. For each NS record we also obtain the first associated A record. On obtaining the NS record, we have the list of authoritative name servers. We compare the sub-domains of *country code TLDs*, *country code second-level domains* and *generic TLDs*. E.g., if the two name servers of a domain are `dns1.p09.nsone.net.` and `ns1.p43.dynect.net.`, then we compare `nsone` and `dynect`. However, we do not only compare the second-level domain names. If there is a third name server for the same domain at `pdns6.ultradns.co.uk.`, then we compare `ultradns` with `nsone` and `dynect`.

To measure the number of domains that use multiple operators, we need to know the owners of the authoritative name servers. Though it is possible to obtain this information from the WHOIS database using the A records we collected, the information obtained does not have a consistent schema and is heavily rate limited [LFS⁺15]. Hence, we use the WHOIS database to only check information for Alexa Top-1k; for the rest of Alexa Top-1m, we take an approach similar to Chung et al. [CvRC⁺17] and rely on the NS records to indicate the DNS operator. We made manual checks to make sure that subsidiaries of large corporations are not classified as separate operators. For instance, Chinese online shopping website `taobao.com` is a subsidiary of the Alibaba group, and we found that one of their name servers is owned by Alibaba and hence, we classified them as the same operator.

If the result of our comparison is that the authoritative name servers are different, then we mark them as different operators. However, if they are the same, we consider the A records as well. If the A records are in different IP address blocks, then we check whether they are owned by the same organization or not. In the case that the IP address is owned by the same organization, then we consider the name servers to be run by the same operator and otherwise, we consider the name servers to be run by different operators. Note that large organizations such as Facebook and Google run dedicated networks that provide DNS redundancy. However, as they are run by the same organization, we do not account for them in our list of domains with multiple operators.

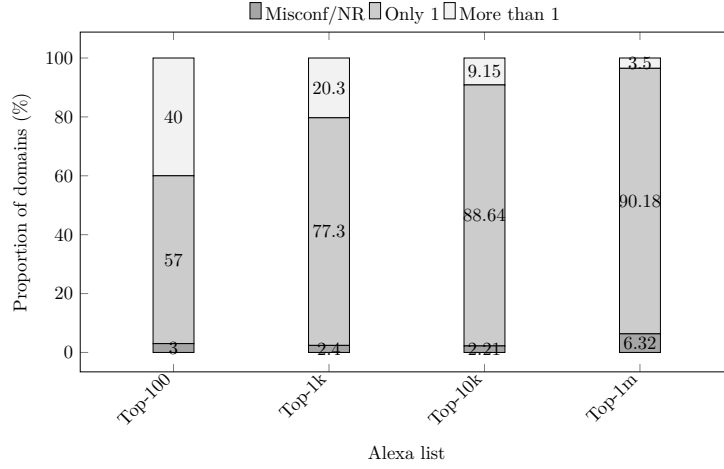


Figure 5.1: Proportion of domains with multiple operators.

5.1.2 Data Analysis

We have classified domains as having a single operator (Only 1), multiple operators (More than 1), no response (NR) and misconfigured (Misconf). An NR classification refers to the case where, during our scans, we did not receive a response from the name server list within a 15-second timeout. Misconf refers to zones that are misconfigured due to mistakes and/or typos. More precisely, we first observed whether we received an A record for the NS record. If we instead receive an error, we then checked the NS record for completeness. If, during this check, we encounter mistakes or typos, the domain is marked as misconfigured. E.g., just `ds0.` was configured as one of the authoritative name servers for the domain `oxfordlearnersdictionaries.com`.

We present our results from the Alexa Global Top 1 million list as well as its subsets in Figure 5.1. We did not receive a response to our queries from 3, 24, 208, 60775 domains in the Top-100, Top-1k, Top-10k and Top-1m respectively. Although we did not find any misconfigured domains in the Top-1k, we found 13 misconfigured domains in the Top-10k and 2483 domains in Top-1m. We observe that 40% of the domains in Alexa Top-100 have more than one operator while the proportion reduces as we move down the Top-1m list. 20.3%, 9.2% and 3.5% of the domains in the Top-1k, Top-10k and Top-1m have multiple operators for their domain. Hence, we conclude from our measurements that there are thousands of domains that use multiple operators and that can easily use our threshold ECDSA protocols.

5.2 System and Threat Model

The diversity of the DNS ecosystem should be reflected in our system and threat model. For the system model, we assume a small number of operators that serve a single domain. As seen in the previous section, this setting is common in practice, in particular among popular domains. For the threat model, we take the two issues—centralization of key management and operation—outlined in the introduction as our starting point. Before we continue, we describe the security properties that we address with our threat model:

1. **Key Privacy.** This is our baseline. Key privacy states that signing key remains private in the event that a server is compromised. We note that this property is relevant in a number of different contexts. For example, this property states that a signing key isn't exposed to a system administrator, or to anyone who obtains a decommissioned (but improperly cleaned) server.
2. **Operational Integrity.** Besides keeping keys secret, we may also want to uphold the integrity of operation. By operational integrity, we mean that only two situations can occur: Either operation proceeds as normal, that is, the right zone is signed, *or* nothing is signed. In other words, at best, a malicious operator can only disrupt operation, i.e., it performs a denial of service attack but it cannot sign zones with bogus information. Notice that key privacy is subsumed in operational integrity. If it is possible to extract the signing key, then no guarantee about the integrity can be made since a single operator can sign any zone it manages.

5.2.1 System and Communication Model

Intuitively, our system model can be viewed as distributing the task of a single operator among multiple operators. We assume that the operators can securely communicate with each other, e.g., using a TLS connection where both endpoints are authenticated. Although a larger pool of operators increases the availability of the domain, domain owners also need to consider the cost-benefit trade-off. As two or three operators provides sufficient redundancy, we envision our system to be used in the setting where the number of DNS operators per domain is $n = 2$ or $n = 3$. These operators can be distributed in a single location, communicating over

a LAN, or they can be distributed globally. Finally, we assume that the servers are sufficiently separated, i.e., a compromise of one server does not automatically lead to a compromise of another server.

5.2.2 Threat Model

We consider an adversary that is capable of compromising a single server. Thus, when $n = 2$, the adversary controls half the servers, and when $n = 3$, the adversary controls a minority (since 2 servers remain honest). We distinguish between two standard adversarial behaviours described in § 2.1. The first type of adversary, called *passive*, follows the prescribed protocol. The second adversary, called *active*, may behave arbitrarily and may not follow the protocol. These two adversarial types capture our security properties. If we only desire key privacy, then security against a passive adversary suffices. If we want operational integrity as well, then we must also secure ourselves against active adversaries. Indeed, it is exactly against such an adversary that the integrity of operation becomes an issue.

5.3 Threshold ECDSA

In this section, we describe ECDSA in § 5.3.1. Next, we describe a standard MPC functionality over fields and then introduce extensions to compute on groups in § 5.3.2. Then, we describe how we provide active security in § 5.3.3. Once we have all the pieces, we describe the complete multiparty threshold ECDSA protocol in § 5.3.4 and its security analysis in § 5.3.5.

5.3.1 ECDSA

For zone signing, the two most widely used signing algorithms are RSA and ECDSA. RSA continues to be widely deployed while the rate of adoption of ECDSA is increasing since it was standardized for DNSSEC in 2012 [HW12, vRJS16]. We focus on ECDSA [KG13], which is parameterized by a subgroup $\mathcal{G} \subseteq E(K)$ of prime order p of some curve $E(K)$, where G is a generator of \mathcal{G} . We use \mathbb{Z}_p to denote a field of order p and H to denote a hash function mapping messages unto elements of \mathbb{Z}_p . We describe the signature scheme that is represented by a tuple $(\text{KGen}, \text{Sig}, \text{Vf})$ where:

- $\text{KGen}(1^\lambda)$
 1. Sample at random $\text{sk} \leftarrow \mathbb{Z}_p$ as the signing key.
 2. Compute the public key $\text{pk} = \text{sk} \cdot G$ as the public verification key.
 3. Output the keys (sk, pk) .
- $\text{Sig}(\text{sk}, M)$
 1. Sample an instance key $k \leftarrow \mathbb{Z}_p$ at random.
 2. Compute $R = (r_x, r_y) = k \cdot G$. If $r_x \equiv 0 \pmod{p}$, go back to Step 1.
 3. Compute $s = k^{-1}(H(M) + \text{sk} \cdot r_x)$.
 4. Output the signature $\sigma = (r_x, s)$.
- $\text{Vf}(\text{pk}, M, \sigma)$
 1. Parse σ as (r_x, s) .
 2. Compute $(r'_x, r'_y) = s^{-1}(H(M) \cdot G + r_x \cdot \text{pk})$.
 3. Output 1 iff $r'_x = r_x$.

We can check that Sig produces a valid signature:

$$\begin{aligned}
& s^{-1}(H(M) \cdot G + r_x \cdot \text{pk}) \\
&= k(H(M) + \text{sk} \cdot r_x)^{-1}(H(M) \cdot G + r_x \cdot \text{pk}) \\
&= k \cdot G \cdot ((H(M) + \text{sk} \cdot r_x)^{-1}(H(M) + \text{sk} \cdot r_x)) \\
&= k \cdot G = (r_x, r_y).
\end{aligned}$$

5.3.2 Secure Computation on Groups

We assume an MPC engine supporting the standard commands of the *arithmetic black-box (ABB)* functionality as described in Figure 5.2, where the notation $[a]$ indicates that the value a is secret-shared, i.e., no party has access to it. The security model of an MPC protocol is parametrized by two parameters. The first parameter is whether the adversary can control at least half, or less than half the parties. The former is called dishonest majority, while the latter is called honest majority. When one server is corrupted, an honest majority protocol would correspond a setting with $n = 3$ servers, while a dishonest majority protocol means $n = 2$. The second

Arithmetic Black-Box

- A command $([a], [b], [c]) \leftarrow \text{RandMul}()$ that generates appropriate representations of a random tuple of secret shared values $a, b, c \in \mathbb{Z}_p$ with $c = ab$.
- A command $[c] \leftarrow \text{Mul}([a], [b])$ that returns $c = ab$. This is typically implemented using one invocation of `RandMul` and Beaver’s re-randomization technique (See § 2.3.2).
- A command $[a] \leftarrow \text{Rand}()$ that generates appropriate representation of a random value $a \in \mathbb{Z}_p$.
- A command $a \leftarrow \text{Open}([a])$ that publicly reconstructs a (or outputs a special symbol \perp denoting abort).
- Linear computation for the $[\cdot]$ representation: given the shares $[a], [b]$ and public scalars $x, y \in \mathbb{Z}_p$, the parties can compute $[c] = x \cdot [a] + y \cdot [b]$ “for free”, i.e., the computation does not involve communicating with the other parties.

Figure 5.2: Arithmetic black-box functionality.

parameter is the corruption model: The two cases—active and passive—correspond to our description in § 5.2.2.

We present an extension to the ABB that extends its capabilities to secure computation over an arbitrary abelian group of order p . In some sense, this shows that the actual representation of the algebraic structure used to perform MPC is irrelevant as long as it is possible to perform linear operations. This generalization of arithmetic MPC has also been described independently by Smart and Talibi Alaoui [SA19], and might have applications in other contexts.

We use this idea to perform MPC in subgroup \mathcal{G} . This extension comes at no extra cost in terms of communication and a small increase in computation complexity corresponding to standard operations in the subgroup of the curve. Consider a protocol implementing the ABB in Figure 5.2 and assume that the shares $[a]$ are elements of \mathbb{Z}_p . The idea is to let each party map their share of $[a]$ to a curve point of order p by locally computing $A_i = a_i \cdot G$, where a_i is party i ’s share of a . This mapping, being a homomorphism, preserves linearity such that A_i is a share of $a \cdot G$ with the same properties as the original \mathbb{Z}_p sharing $[a]$. Let $\langle a \rangle$ denote a share of

$a \cdot G$. Then, we add the following two commands to the ABB in Figure 5.2:

- A command $\langle a \rangle \leftarrow \text{Convert}([a])$ that converts a representation of the shared value a in \mathbb{Z}_p to a representation of the value $a \cdot G$ in the group \mathcal{G} .
- A command $a \cdot G \leftarrow \text{Open}(\langle a \rangle)$ that recovers the secret shared point.

These two commands, along with the functionality of the original ABB, which provide secure computation over \mathbb{Z}_p , are sufficient to give us a protocol for secure computation over the group \mathcal{G} . If we consider the sharing $[a]$ as a vector with elements from \mathbb{Z}_p , we get the following useful properties:

- Linearity is preserved, i.e., given the shares $\langle a \rangle, \langle b \rangle$ and scalars $x, y \in \mathbb{Z}_p$, we can locally compute $\langle c \rangle = x\langle a \rangle + y\langle b \rangle$.
- If the **Open** procedure for $[\cdot]$ shares relies only on group operations in \mathbb{Z}_p , then we can implement **Open** for $\langle \cdot \rangle$ shares by using the corresponding group operations of \mathcal{G} . This property follows from **Convert** being structure preserving.
- Secret scalar multiplication by public point is possible by noting that **Convert** defines an action of \mathbb{Z}_p on \mathcal{G} , i.e., $[a] \cdot P$ for a $P \in \mathcal{G}$ is a local operation that results in $\langle a \cdot \log_P(G) \rangle$. Note that opening this share will result in $a \cdot P$.
- Finally, given $[x], \langle y \rangle$ and a multiplication tuple $[a], [b], [c]$, it is possible to compute $\langle xy \rangle$ tweaking Beaver's circuit randomization (see § 2.3.2) as follows:
 1. $e = \text{Open}([a] + [x])$,
 2. $D = \text{Open}(\text{Convert}([b]) + \langle y \rangle)$, and
 3. $\langle xy \rangle = \text{Convert}([c]) + e\langle y \rangle + [x]D - eD$.

Note that the final property is not required for the applications in this thesis, but it could be of independent interest.

The properties of **Convert** and **Open**, as well as the functionality of the underlying ABB that provides secure computation over \mathbb{Z}_p is sufficient to give us a protocol for secure computation over \mathcal{G} . This extended ABB, which we will call ABB+, is described in Figure 5.3.

Extended Arithmetic black-box

- $\text{RandMul}()$, $\text{Mul}([\cdot], [\cdot])$, $\text{Rand}()$, $\text{Open}([\cdot])$ as described in Figure 5.2.
- A command $\langle a \rangle \leftarrow \text{Convert}([a])$ that converts a representation of a secret $[a]$ over the field \mathbb{Z}_p into a representation of the secret $\langle a \rangle$ over the group \mathcal{G} .
- A command $a \cdot G \leftarrow \text{Open}(\langle a \rangle)$ that reconstructs a curve point $a \cdot G$ from a secret representation $\langle a \rangle$.

Figure 5.3: Extended arithmetic black-box.

5.3.3 Active Security using SPDZ like MACs

In the previous section, we showed that we can easily extend a protocol of \mathbb{Z}_p with functionality for secure computation over a subgroup of $\mathcal{G} \subseteq E(K)$ of order p . A natural question to ask is whether the active security guarantees of the \mathbb{Z}_p protocol extend to the \mathcal{G} protocol. We answer this question in the affirmative by showing that the MAC scheme of SPDZ [DPSZ12], which is unconditionally secure, can be used to provide authentication of shares in \mathcal{G} (i.e., $\langle \cdot \rangle$ shares) as well.

SPDZ recap. We recall the SPDZ protocol and its security using the description from Damgård et al. [DKL⁺12]. In SPDZ, a value $a \in \mathbb{Z}_p$ is shared as

$$[a] = ((a_1, \dots, a_N), (\gamma(a)_i, \dots, \gamma(a)_N)),$$

where party i holds the pair $(a_i, \gamma(a)_i)$, and where $a = \sum_i a_i$ and $\alpha \cdot a = \gamma(a) = \sum_i \gamma(a)_i$. The value $\alpha \in \mathbb{Z}_p$ is a global MAC key that is secret shared using a different scheme, $[[\alpha]]$. The details of this are not important for the following discussion; it suffices to say that each party has a share α_i , such that $\sum_i \alpha_i = \alpha$, as well as other information to make this sharing secure. The global MAC key is unknown to all parties and provides a notion of authentication of the shares.

We recap the opening phase of the SPDZ protocol for a single value, i.e., the part where the parties check if the output was computed correctly ²:

1. Each P_i has input α_i , their share of the global MAC key, and $\gamma(a)_i$, their share of the MAC on a partially opened value a ³.

²Several openings can be batched. see the original paper [DKL⁺12] for more details.

³A partial opening reveals the value but not the MAC.

2. Each P_i computes $\tau_i = \gamma_i(a) - \alpha_i a$ and broadcasts a commitment $\text{com}(\tau_i)$.
3. All parties open $\text{com}(\tau_i)$, and compute $\text{chk} = \sum_i \tau_i$. If $\text{chk} \neq 0$, output \perp and abort.

Suppose $a' = a + \epsilon$, i.e., the adversary adds an error $\epsilon \neq 0$ during the partial opening. In addition, if the adversary lies about its MAC in Step 2 of SPDZ opening phase, an error that we denote as Δ , then the adversary is successful if $\Delta = \sum_i \tau_i$. In this case, we have

$$\Delta = \sum_{i=1}^n \tau_i = \sum_{i=1}^n \gamma_i(a) - \alpha_i a = \alpha \epsilon.$$

Since $\epsilon = (a - a') \neq 0$, then $\alpha = \Delta \epsilon^{-1}$, which happens with probability at most $1/p$ due to the random choice of α .

SPDZ-like computation over an elliptic curve. In the remainder of this section, we will use the shorthand notation $\text{cv}(a) = \text{Convert}(a)$ interchangeably for convenience. Consider the most natural modification possible to obtain a notion of a SPDZ-sharing $\langle \cdot \rangle$ over \mathcal{G} from a SPDZ-sharing $[\cdot]$ over \mathbb{Z}_p by applying cv to all local shares. We define $\langle a \rangle$ as the vector

$$\langle a \rangle = ((\text{cv}(a_1), \dots, \text{cv}(a_N)), (\text{cv}(\gamma(a)_1), \dots, \text{cv}(\gamma(a)_N))),$$

where P_i holds $(\text{cv}(a_i), \text{cv}(\gamma(a)_i))$. Observe that the linearity of cv implies that

$$\sum_i (\text{cv}(a_i)) = \text{cv}(\sum_i a_i) = \text{cv}(a),$$

which makes $\langle a \rangle$ a valid sharing of $\text{cv}(a)$. In addition, the semantics of the MAC is preserved since

$$\sum_i \text{cv}(\gamma(a)_i) = \text{cv}(\sum_i \gamma(a)_i) = \text{cv}(\alpha \cdot a).$$

Therefore, we can use the same $\llbracket \alpha \rrbracket$ to authenticate the **Converted** share as well. More precisely, we consider a modified opening procedure that works as follows⁴:

1. Let α_i be the share of the key held by P_i , and $\Gamma_i = \text{cv}(\gamma(a)_i)$ be the shares of the MAC on $A = \text{cv}(a)$.

⁴We describe the procedure for a single value. It can be extended to support batching.

2. Each P_i computes $\Sigma_i = \Gamma_i - \alpha_i A$ and broadcasts a commitment $\text{com}(\Sigma_i)$.
3. Open $\text{com}(\Sigma_i)$, compute $\text{chk} = \Sigma_1 + \dots + \Sigma_N$. If $\text{chk} \neq 0$, output \perp and abort.

Due to the linearity of the group operations, if the adversary opens $A' \neq A$, then the check only passes with probability $1/p$. In a nutshell, we are taking a secure linear MAC procedure, and raising all the MACs and values in the exponent.

5.3.4 Multiparty ECDSA Protocol using ABB+

We recall the protocol of Gennaro and Goldfeder [GG18] and show that it can be computed by our extended arithmetic black box functionality. The main issue with computing ECDSA signatures securely is calculating k^{-1} such that it does not reveal information about k . However, the inversion trick by Bar-Ilan and Beaver [BB89] can be used here:

1. Suppose each party has a share of two random values b, k and their product, i.e., $([b], [k], [c])$ where $c = b \cdot k$.
2. The parties $\text{Open}([c]) = c$.
3. They locally compute $c^{-1}[b] = [(k \cdot b)^{-1}b] = [k^{-1}]$

Thus the price to pay for the inversion, which is the most expensive part of every threshold ECDSA protocol, is essentially generating a random multiplication triple using RandMul , and using Convert to compute the value $R = \text{Open}(\text{Convert}([k]))$. The other value we need is a sharing of sk/k . Given $[k^{-1}]$ it is possible to compute $[\text{sk}/k]$ very efficiently by performing a single secure multiplication.

The full protocol using the ABB+ now follows: We consider a setting with a number of servers $\mathcal{S} = \{S_1, \dots, S_N\}$ and a number of users $\mathcal{U} = \{U_1, \dots, U_\ell\}$. Our protocol has four phases:

1. We generate a random secret key using $[\text{sk}] = \text{Rand}()$, and then use it to generate the public key by running $\text{pk} = \text{Open}(\text{Convert}([\text{sk}]))$ (Alternatively, users can pick their own keys and input them to the servers in \mathcal{S}).
2. We have two preprocessing phases. The first preprocessing phase is independent of the users and the messages to be signed, and serves to generate the values $[k^{-1}]$ and $\langle k \rangle$ that are required for generating any signature;

3. The second preprocessing phase depends on the user and computes $[\text{sk}_j/k]$, where sk_j is the signing key of user U_j .
4. Finally, generating a signature using the output of the preprocessing and the user's signing key involves a linear computation followed by an opening.

5.3.5 Security Analysis

We show the details of the full protocol in Figure 5.4. The security of the protocol follows directly from the security of the underlying ABB scheme, and from the assumption that ECDSA is a secure signature scheme. This assumption has also been used by Doerner et al. [DKLS18, DKLS19].

It is possible to open R in the user independent stage of the protocol in Figure 5.4 before the message is known. However, in that case, the security does not directly reduce to the security of ECDSA as the adversary is allowed to choose its message dependent on the nonce R . We can handle such an adaptive adversary such that the reduction will try to guess which of the nonces the adversary will use, and act accordingly. If the guess is correct, then the reduction proceeds as in the non-adaptive case. Otherwise, it fails. Since the adversary can generate only a polynomial number of nonces, security is still preserved, but the reduction is not tight anymore. Canetti, Makriyannis and Peled [CMP20] treat this topic in more detail.

5.4 Multiparty Zone Signing System

In this section, we describe the integration of our threshold ECDSA implementation in a DNS name server before describing the important operations. We implement several variants of our threshold ECDSA protocol on top of MP-SPDZ [Dat19] and have used Crypto++ as the library for computation over elliptic curves. We integrate MP-SPDZ with DNS administrative name servers. For DNS name server software, we use Knot DNS [Kno19] as it has the possibility to perform automated key management and it comes with extensive documentation. For the setting where the registrar is the DNS operator, we propose that registrars interact with other registrars in the zone signing protocol. We describe the multi-operator setting in this section and, where necessary, we note the difference if the operators are also the registrar.

Threshold ECDSA in the ABB+ Hybrid Model

Key Generation. $([sk_j], pk_j) \leftarrow KGen(1^\lambda)$

To generate a key for user U_j , either U_j supplies the sharing $[sk_j]$, or the servers run $[sk_j] \leftarrow \text{Rand}()$. The public key is computed as $pk_j = \text{Open}(\text{Convert}([sk_j]))$.

User independent preprocessing. $(\langle k \rangle, [k^{-1}]) \leftarrow \text{PInd}()$

1. The servers run $([a], [b], [c]) \leftarrow \text{RandMul}()$.
2. Run $c \leftarrow \text{Open}([c])$.
3. Let $[k^{-1}] = [a]$.
4. Define $\langle k \rangle \leftarrow \text{Convert}([b]) \cdot c^{-1}$.
5. Output $(\langle k \rangle, [k^{-1}])$.

User dependent preprocessing. $(\langle k \rangle, [k^{-1}], [sk'_j]) \leftarrow \text{PDep}([sk_j], \langle k \rangle, [k^{-1}])$

1. Take as input $[sk_j]$ (the sharing of the secret key of user U_j) and $(\langle k \rangle, [k^{-1}])$ (an unused tuple from the previous phase).
2. Compute $[sk'_j] = [sk_j/k] \leftarrow \text{Mul}([k^{-1}], [sk_j])$.
3. Output a final tuple $(\langle k \rangle, [k^{-1}], [sk'_j])$.

Signing. $\sigma \leftarrow \text{Sig}(\langle k \rangle, [k^{-1}], [sk'_j], M)$

1. Take an unused preprocessed tuple $(\langle k \rangle, [k^{-1}], [sk'_j])$ for U_j and the message M to be signed.
2. Run $R \leftarrow \text{Open}(\langle k \rangle) = (bc^{-1}) \cdot G = a^{-1} \cdot G = k \cdot G$.
3. Let $(r_x, r_y) \leftarrow R$.
4. Compute $[s] = H(M) \cdot [k^{-1}] + r_x \cdot [sk'_j]$.
5. Open $s \leftarrow \text{Open}([s])$ and output the signature $\sigma = (r_x, s)$ or $\sigma = \perp$.

Figure 5.4: Full protocol for threshold ECDSA signatures using ABB+.

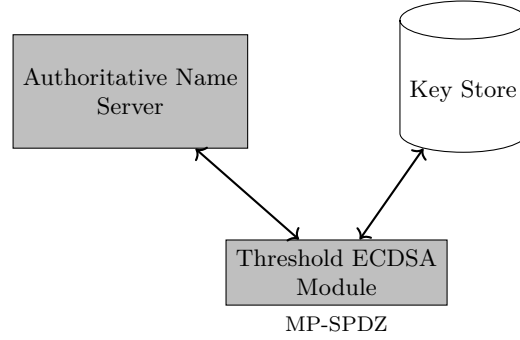


Figure 5.5: Setup at DNS operator.

5.4.1 Setup

Figure 5.5 shows the architecture of the proposed DNSSEC signing system at a DNS operator. In our DNSSEC signing system, each operator serves a name server, runs a threshold ECDSA module and has two key stores: one to store the keys for particular zones and another to store the key material associated with other operators. We consider three name servers operated by independent DNS operators, all of which support ECDSA with SHA256 message digest. We do not change the operation of Knot DNS apart from the parts involved in DNSSEC key generation, key rollover and zone signing. Communication between the name server and the threshold ECDSA module is performed using a message queue.

We note that although we describe our automated system where the DNS operators handle key generation/rollover in addition to zone signing, it is possible to use our system in the setting where the domain owner handles the key generation/rollover. In that case, the only change would be that the domain owner generates the key pair and the private key shares and sends them to the DNS operators.

5.4.2 Key Generation/Rollover

In the key generation/rollover phase, when new keys need to be generated, each operator generates a signing key share $[sk_j]$ for the zone and runs the key generation as shown in Figure 5.4. At the end of this phase, the public key is added to `DNSKEY` record of the zone at all the operators and the signing key share $[sk_j]$ is stored in the keystore for the zones. In addition, a tag that indicates the DNS operators associated with this signing key share is stored. E.g., Operator A would store a

tag $T(B, C)$ along with the key shares associated with Operator B and Operator C. This makes it easy for the threshold ECDSA module to contact the corresponding DNS operators during the signature generation phase. Note that the key generation for ZSK and KSK is the same except that in the case of KSK, the domain owner generates the DS record and sends it to the registrar, who then submits it to the registry. When the registrar is one of the DNS operators of the zone, then the registrar can directly submit the DS record.

5.4.3 Signing

As shown in Figure 5.4, our signing protocol has three phases: the first is independent of the zone to be signed, while the second is independent of the RRset, but dependent on the zone to be signed. Each of the three phases involve the three steps that are shown in Figure 5.6. In Step 1, the threshold ECDSA module receives the input for the phase from the name server and the tag from the key store. In Step 2, the MPC protocol for the phase is run between the threshold ECDSA module of the three operators. In Step 3, the output of the preprocessing phases are sent to the key store (Step 3p) while the output of the signing phase, **RRSIG**, is sent to the name server to store in the zone file (Step 3s). We note that the threshold ECDSA module runs in the background and periodically polls the name server so that it is always available to sign.

Accountability of DNS operators. When the DNS operators serve a domain independently, it is straightforward to identify the operator in the case of malfunction. When we use our multiparty zone signing system, accountability requires a few extra checks. The protocols we implement are in the security-with-abort model. In the case of honest majority protocols, we can detect the malicious behaviour by local checking. As two operators are sufficient to generate the final signature, they can check which party sent corrupted inputs in case the protocol fails with the third party. This mechanism only works in the online phase. For the preprocessing phase, we need more checks that requires us to instantiate with protocols in identifiable-abort model. Similarly, a protocol in identifiable-abort model is required for dishonest majority and it makes auditing by an external party possible as well. Although there are black-box extensions from abort to identifiable-abort protocols, these are not yet efficient [CHOR18].

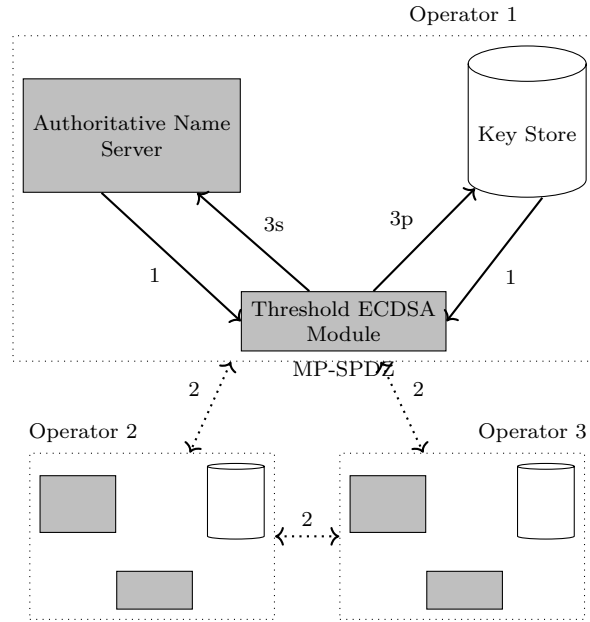


Figure 5.6: Zone signing.

Implication for DNS Operators. In our system, the DNS operators do not need to be online any more than they already are in existing systems. DNS operators in existing systems remain online to respond to DNS queries. Many DNS operators sign DNS responses on-the-fly and, hence, they are already equipped with signing systems that are online. In our system they will not only respond to DNS queries, they will also run MPC with other registrar/operators to create RRSIG. Our threshold ECDSA protocols have an overhead—both in terms of communication and computation—that depends on the concrete threat and system model. We discuss the overheads as part of our benchmarks in § 5.5. It is also worth noting that the operators need not rely on secure hardware to store their user’s keys anymore, which may bring down both the cost and complexity for a DNS operator.

Implication for DNS resolvers. Proper functioning of the DNSSEC ecosystem requires both the signing and the validation to work. Deploying changes at DNS resolvers, who perform validation, is extremely hard as numerous resolver software need to be changed. Fortunately, no change is required at the validating resolver to use our solution. Every time the domain is queried at the authoritative name server, the signatures for the zone need to be verified at the resolvers for the chain of trust

to be established. Though three operators are involved in the signing process, the signature can be verified with the same `DNSKEY`, irrespective of the operator which initiated the signing process. If the DNS resolver obtains the `DNSKEY` records from Operator A and stores it in the cache, then it will be able to authenticate a response from Operator B for the same domain, as the two operators have the same `DNSKEY` for the zone. The resolver will be able to verify the chain of trust irrespective of the operator responded to the query.

5.5 Evaluation

In this section, we report on several benchmarks of our protocol and compare with prior work of both signature generation and key generation times. We implement six instantiations of our protocol, which support different system and threat models, in MP-SDPZ [Dat19]. For $n = 3$ we have *Rep3*, *Shamir* (passive security) and *Mal. Rep3* and *Mal. Shamir* (active security). For $n = 2$, we use *MASCOT* and *MASCOT-* (*MASCOT* minus). We discuss these protocols in § 5.5.1.

We benchmark our implementation with $n = 2$ and $n = 3$, depending on whether the underlying protocol requires an honest majority or not. In our benchmarking, we split the protocol between preprocessing (including both preprocessing phases in Figure 5.4) and online signing phase. We measure both the throughput (based on batches of 10000 signatures), and the time for generating a single signature given a preprocessed tuple. Many of these protocols have asymmetric communication patterns and, thus, we report the maximum execution time, instead of the average.

Experimental setup. We used VMs from AWS `c5.2xlarge` in three settings:

LAN setting or **Colocation:** All servers located in the same region in Ireland.

WAN setting: The servers are located in two sets of locations.

Continent: Servers are located in the same continent. We used servers in Ireland, Paris and London. For two-party protocols, benchmarks were run between Ireland and Paris.

World: Servers are located in different continents. We used one server each in Ireland, North Virginia and Seoul. For two-party protocols, the servers in

Ireland and Seoul were used. The largest RTT between any two servers in the world setting is approximately 240ms.

RTT for our setup can be found in Table 5.1. All benchmarks were run with a single thread.

5.5.1 Protocols

As we discussed in § 2.3.2, we use replicated secret sharing and Shamir’s secret sharing for honest-majority protocols. Rep3 and Mal. Rep3 are replicated secret sharing-based three party protocols in semi-honest and malicious security models respectively. Shamir and Mal. Shamir are Shamir’s secret sharing based protocols in semi-honest and malicious security models respectively. While we benchmark the three-party variant of these protocols, these protocols can be use for $n > 3$. MP-SDPZ implementation supports $n > 3$ for Shamir. Replicated secret sharing does not scale well for more parties as the size of shares increases exponentially with the number for parties [FLNW17]. Hence, only the three-party variant of replicated secret sharing based protocol is implemented in MP-SPDZ.

For the dishonest majority protocols, we use MASCOT and MASCOT-. MASCOT [KOS16] is an efficient OT-based preprocessing method for SPDZ protocols. It includes checks to detect attacks and it is secure against malicious adversaries. MASCOT- is an optimization of MASCOT. We describe MASCOT- in § 5.5.2.

5.5.2 MASCOT- Optimizations

Our MASCOT- protocol is obtained by making a number of function specific optimizations to MASCOT [KOS16]. Threshold signatures are a special case of MPC where the correctness of the output can trivially be determined by observing the output itself, by verifying the signature. This is a well known trick which has previously been used to optimize many threshold ECDSA protocols in the literature. We can similarly optimize our protocol by using an “optimistic” version of the `Open` command when running Step 3 of the *Signing* subroutine in Figure 5.4.

SPDZ opening. We save a round of communication while opening as we do not need to check correctness of the MACs. An adversary might attempt an additive

	AB (ms)	AC (ms)	BC (ms)
Colocated	0.07	0.07	0.08
Continent	11.51	16.98	8.2
World	240.0	71.0	181.1

Table 5.1: RTT in milliseconds between the servers used. A is always Ireland, while B is Paris and Seoul in the “Continent”, respectively “World” setting, and C is London and N.Virginia in the “Continent”, respectively “World” settings.

attack, which may result in an invalid signature. However, the protocol does not leak any information about the secret key.

Beaver multiplication. Suppose the adversary can perform an additive attack during multiplication, i.e., $x + a + \epsilon_1$ and $y + b + \epsilon_2$ for independent ϵ_1 and ϵ_2 . A multiplication becomes

$$(x + a + \epsilon_1) \cdot (y + b + \epsilon_2) - (x + a + \epsilon_1) \cdot b - (y + b + \epsilon_2) \cdot a + ab = xy + \epsilon_1 y + \epsilon_2 x + \epsilon_1 \epsilon_2.$$

This permits a selective failure attack. E.g., $\epsilon_2 = 0$, $\epsilon_1 \neq 0$ then the multiplication is correct if and only if $y = 0$. However, multiplications are only used on k^{-1} and sk , both of which are of high entropy.

5.5.3 Comparison with Prior Work

We present a comparison of our protocols with two industry protocols from Unbound [Unb19] and KZen [KZe19], as well as the two-party protocol of Doerner et al. [DKLS18] (DKLS) in Table 5.2. The numbers reported for our protocols correspond to running all three phases in Figure 5.4. We see that MASCOT- performs as well as the fastest prior protocol in DKLS, with the same security guarantees, in the LAN setting. However, with more servers, some of our protocols perform better in the LAN setting. In our two WAN settings, DKLS outperforms our protocols as it requires only 2 messages (1 round of communication) whereas our fastest protocol (Rep3) requires 3. Interestingly, the simplicity of our key generation protocol is apparent, and in all cases (except MASCOT-) key generation is faster than signing.

5.5.4 Key Generation

We also benchmark the key generation phase as that is typically the more expensive phase in prior works [GG18, LNR18]. With our approach, generating a shared key amounts to running any protocol for generating a secret shared field element $[\text{sk}]$, followed by opening the result of $\text{Convert}[\text{sk}]$. Timings for key generation are shown in Table 5.3. For our honest majority protocol ($n = 3$), generating a secret key requires only 1 or 2 rounds of communication. The opening procedure of MASCOT and MASCOT $^-$ is more costly than the rest of the protocols. The heuristics used to obtain MASCOT $^-$ cannot be used when generating keys and, hence, the key generation time is about the same for both.

5.5.5 Amortizing Signing

Finally, we analyze the cost of signing when amortization is applied, something that no prior work has considered ⁵. Table 5.4 shows how many signing tuples each protocol can generate per second. The signing times reported in this table correspond to computing a signature when amortization is taken into account. A signing tuple corresponds to the output of the *user dependent preprocessing* phase in Figure 5.4. We note that, for almost all protocols, amortized signing corresponds essentially to a single round of communication.

5.5.6 Overhead for Operators

The storage overhead can be derived from the sizes of a share for a given protocol. For Mal. Rep3, MASCOT and Rep3 each share consists of two \mathbb{Z}_p elements, while for the rest a share is a single element. Thus, for the former three the overhead for storing the signing keys is doubled. A signing tuple consists of two \mathbb{Z}_p shares and one \mathcal{G} share. For example, Rep3 needs to store roughly $2 \cdot 4 \cdot 32$ bytes per signature, assuming a 256-bit prime. Communication per party is between 177 and 354 bytes, depending on the protocol (this number was derived experimentally).

⁵Although it might be possible to split some of the protocols in previous work into a preprocessing and signing phase, such a split has not been implemented and, hence, we cannot compare with them.

	n	Colocation		Continent		World	
		Sig(ms)	KGen(ms)	Sig(ms)	KGen(ms)	Sig(ms)	KGen (ms)
Rep3	3	2.78	1.45	27.22	29.44	367.87	291.32
Shamir	3	3.02	1.39	78.75	35.52	1140.09	486.82
Mal. Rep3	3	3.45	1.57	82.14	39.97	1128.01	429.47
Mal. Shamir	3	4.43	1.89	174.95	37.35	2340.53	485.11
MASCOT	2	6.56	4.32	196.19	185.71	2688.92	2632.07
MASCOT-	2	3.61	4.41	54.38	181.12	729.08	2654.59
DKLS [DKLS18]	2	3.58	43.73	15.33	109.80	234.37	1002.97
Unbound [Unb19]	2	11.33	315.96	31.08	424.02	490.73	1010.98
Kzen [KZe19]	2	310.71	153.87	1282.81	577.67	14441.83	7237.93

Table 5.2: Comparison of our threshold ECDSA protocols with prior work. Runtime of our protocols has been obtained by taking the mean over the maximum execution time over many runs.

	Colocation		Continent		World	
	Secret (ms)	Public (ms)	Secret (ms)	Public (ms)	Secret (ms)	Public (ms)
Rep3	0.16	1.27	11.12	18.31	113.86	174.03
Shamir	0.25	1.13	17.17	18.09	243.00	243.82
Mal. Rep3	0.16	1.40	11.00	28.98	115.25	301.66
Mal. Shamir	0.25	1.62	16.90	18.32	241.78	243.18
MASCOT	2.34	1.91	149.26	33.01	2142.31	442.75
MASCOT-	2.40	1.92	145.48	33.21	2132.75	449.43

Table 5.3: Breakdown of key generation benchmarks into the time it takes to generate the $[sk]$ sharing, and the time it takes to run $\text{Open}(\text{Convert}([sk]))$. Runtimes are the maximum time that each step takes.

5.6 Related Works

DNSSEC deployment and measurement. DNSSEC deployment heavily relies on DNS operators and registrars. Prior works have found issues such as reuse of signing keys by DNS operators for multiple domains [CvRC⁺17] and sharing of RSA modulus among multiple domains [SW17]. Additionally, a large fraction of domains sign their DNSKEY record twice: once with the KSK (as expected) and once with the ZSK (which is not used in validation) [CvRC⁺17]. This not only increases the DNSKEY packets sizes, but could lead to fragmentation attacks (if strong RSA keys of size 2048 bits or more are used). This makes domain resolution inefficient [VDBvRDSP14], but also makes DNSSEC vulnerable to poisoning attacks

	Colocation		Continent		World	
	Tuples per sec.	Sig (ms)	Tuples per sec.	Sig (ms)	Tuples per sec.	Sig (ms)
Rep3	922.27	2.49	898.25	19.91	715.54	247.13
Shamir	1829.69	2.37	1544.31	20.62	402.88	271.80
Mal. Rep3	914.65	2.52	806.13	20.07	309.76	245.14
Mal. Shamir	1792.30	2.91	1154.30	27.03	172.87	416.60
MASCOT	380.19	4.82	233.73	57.02	31.98	756.34
MASCOT-	700.94	2.75	447.85	20.37	68.31	258.85

Table 5.4: Throughput for preprocessing (tuples per second) and the signing time when amortization is taken into account.

when resolvers do not validate responses [HS13a]. After the DDoS attacks of 2016, the impact of the attacks and the number of customers of DyN and NS1 that added another operator was measured [AvRN18]. However, only the domains that use DyN and NS1 were measured while we measure the use of multiple operators, not restricting our measurements to managed DNS providers.

Privacy in DNS. Though DNSSEC provides data integrity, it does not provide confidentiality. “Range queries” [ZHS07a] and private information retrieval [ZHS07b] have been proposed as a solution to hide queries. Herrmann et. al [HMF14] analyse the privacy offered by [ZHS07a] and conclude that in the scenario of web-surfing, inter-related DNS queries are issued by the client and this relation provides a much greater information than single queries. Castillo-Perez and García-Alfaro [CG09] also claim that the above described protocol does not provide sufficient privacy. Hence they proposed to use m servers instead of two and they do not require the servers to remain non-colluding.

The Internet Engineering Task Force (IETF) has made privacy an important topic since the Snowden revelations and protecting DNS information has since been recognized widely. One of the reasons is the understanding that an attack may not always take an active form but can also take the form of pervasive monitoring [FT14]. It has considered privacy issues in DNS and DNSSEC in RFC 7626 [Bor15] and in RFC 7816 [Bor16], and it has standardized DNS-over-TLS in RFC 8310 [DGR18] and DNS-over-HTTPS in RFC 8484 [HM18] to protect DNS queries in transit between the user and resolvers. While privacy of DNS queries has been considered by prior works, we address the issue of privacy of DNSSEC keys.

Threshold Signatures. Multiple proposals for efficient threshold ECDSA protocols suitable for both 2 and n parties have been developed in the recent past. In the 2-party setting, Lindell [Lin17] provides a protocol that achieves a throughput of over 100 signatures per second when using four threads. Doerner et al. [DKLS18] propose a protocol that relies heavily on OT and requires more communication than the protocol of Lindell, making it less efficient in low-bandwidth environments. In the more general n -party setting, Gennaro, Goldfeder and Narayanan [GGN16] present a protocol with a threshold of $t \leq n - 1$, whose communication was reduced by Boneh, Gennaro and Goldfeder [BGG17]. Both these works require a distributed Paillier key generation scheme, which is inefficient. Gennaro and Goldfeder [GG18] present a protocol that does not require a dealer for key generation. Lindell, Nof and Ranellucci [LNR18] present a protocol whose key generation is significantly slower than signing. Finally, Doerner et al. [DKLS19] generalize their 2-party protocol [DKLS18] to work for any number of parties. All these works are protocols tailored for threshold ECDSA in a particular threat model, and hence, restricting their use. Our work uses generic MPC to construct threshold ECDSA protocol and is secure in different threat models.

Smart and Talibi Alaoui [SA19] independently identified the generalization of arithmetic MPC. Both works show how SPDZ-style MACs can be adapted to elliptic curve computation, and notice that the same idea can be used for other arithmetic MPC protocols (e.g., Shamir-secret sharing and replicated-secret-sharing). Both works mention threshold ECDSA as a natural application, while we in addition split the protocol in user-independent preprocessing and user/message-dependent computation, as we imagine many users outsourcing their signing capabilities to the same subset of semi-trusted servers. The main differences are: we have implemented and benchmarked the resulting protocols against state of the art threshold ECDSA protocols, and we demonstrate how this can be effectively integrated into existing DNS software and show a prototype system we deployed. The work of Smart and Talibi Alaoui does not report on any implementation.

After we showed that most of the material required to generate an threshold ECDSA signature can be generated in the preprocessing phase, this idea has been used in follow-up works [GG20, CMP20, DJN⁺20]. Threshold RSA signatures for DNSSEC have been considered in the past. Cachin and Samar[CS04] proposed a distributed DNS to avoid single point of failure, which provides fault tolerance and

security in the presence of corrupted servers. Cifuentes et al. [CHM⁺16] emulate a hardware security module (HSM) at an authoritative name server and they report timings on a LAN which range from tens to hundreds of milliseconds on commodity hardware. Both used RSA threshold signature scheme of Shoup [Sho00].

Distributed RPKI

Resource Public Key Infrastructure [LK12] is a cryptographic method to secure inter-domain routing against prefix and sub-prefix hijacks. It is also a prerequisite for Border Gateway Protocol Security [LS17]. In RPKI, RIRs allocate IP prefixes and authorize specific ASes to be the origin of routes. This information is stored in an ROA. Routers use the ROAs to distinguish legitimate routes from leaked or hijacked routes. This is known as route origin validation (ROV).

The insecurity of inter-domain routing and the ability of RPKI to address the insecurity has not transpired into wide-scale deployment of RPKI [GCH⁺17, HHSW18]. One of the reasons is the possibility of RIRs to unilaterally takedown IP prefixes, either deliberately or accidentally, that will result in the prefix of the affected ASes being unreachable when ROV is performed [CHB⁺13, KM17]. The hierarchical structure of RPKI gives RIRs the power to revoke and invalidate any object that it has issued.

As centralized authorities are easy targets for legal surveillance and coercion, is it possible to prevent a state-sponsored attacker from imposing its demands on RIRs without drastically changing the structure of RPKI? The RIRs are bound by the law of the country they are based in. Their members, however, are based in different countries and do not have a recourse when their prefix is taken down.

In the past, there have been situations where these problems have taken practical relevance. In 2011, RIPE NCC took the state of Netherlands to court when the Dutch police ordered to it to lock registration of four IP address blocks [RIP11, RIP13]. Nevertheless, it was forced to lock down the registrations. More recently, RIPE NCC mistakenly deleted 2669 ROAs on 1 April 2020 and the ROAs were reinstated

on 2 April [RIP20a]. This meant that the announcement for these resources were ‘unknown’. On the day when these ROAs were missing, RosTelecom had a route leak [Qra20]. While the two events seem independent, according to RIPE NCC, 12 prefixes whose ROAs were deleted were affected by the route leak. Furthermore, RIPE NCC transferred an IP prefix block from a member to another entity based on a German court order transferred to them through a Dutch court [RIP20b]. As a matter of procedure, they will do the same if similar situations arises in the future. In the context of RPKI, this means RIPE NCC will “revoke any certificates generated by the RIPE NCC” [RIP19].

In this chapter, we address these issues that are prevalent in the deployed RPKI system by constructing a distributed RPKI system that relies on threshold signatures, a specific instance of MPC. Our solution, without requiring significant changes to BGP and RPKI, restricts the power of RIRs and only allows revocation of allocated resources in legitimate cases with the cooperation of a number of RIRs.

Significance of the threat model

Without RPKI, BGP operates in a default-accept mode where any AS can announce a BGP route for any IP prefix and the other ASes will accept the route by default. The default-accept mode makes BGP vulnerable to prefix hijacks and sub-prefix hijacks [BFZ07, BFMR10, HRA11, DYN08, Cow10]. In a prefix hijack, a malicious AS announces a route for IP prefixes it does not own such that the traffic for those prefixes are sent to it, and in a sub-prefix hijack, a malicious AS announces a more specific IP prefix than the one that has been allocated.

RPKI relies on hierarchical and centralized authorities to be honest. Malfunction or coercion by law enforcement authorities is not incorporated into the threat model. Such a weak threat model creates an imbalance of power between the RIRs and its members. Moreover, the power imbalance with RPKI is greater than with Web PKI. In RPKI, there is no option to request certificates from different authorities, which is possible with Web PKI. Hence, the reliance on specific RIRs is greater.

Members are further weakened when the authority is based in a different country than their own. The manipulations at the level of BGP is more coarse-grained than domain name seizures as BGP granularity is limited to /24, i.e., 256 IPv4 addresses [CHB⁺13]. The RIRs are bound by the law of the country they are based in. If members are affected, they may need to take the issue up in another country.

The slow process may result in the loss of business.

Threshold signatures for RPKI

We propose a distributed RPKI system based on threshold signatures. Threshold signatures provide a method to distribute trust and they are practical in settings where the number of participating parties is small. We consider the setting of *hosted* RPKI, where INR holders are most vulnerable to attacks as they have no control. There are five RIRs—AFRINIC, APNIC, ARIN, LACNIC and RIPE NCC—and threshold signature protocols are practical when there are only five participating parties. Hence, our system requires a threshold of them to agree before making changes to RPKI objects. This prevents any RIR from unilaterally making changes.

Our solution can be described as follows: threshold signatures use shares of the private key, where each of the five RIRs will have a share of the private key and none of them have the entire private key. Using only the shares, the RIRs can collaboratively sign ROAs and CRLs. Our mechanism prevents them from unilaterally acting maliciously. Most importantly, threshold signatures support a stronger threat model where corrupted RPKI authorities are not entirely trusted and yet play a significant role in making BGP secure.

Contributions

A summary of our contributions:

- We construct a distributed RPKI system based on threshold signatures that addresses three issues: (1) preventing unilateral IP prefix takedowns, (2) limiting the scope and implications of attacks on RIRs, and (3) enabling validation in the case of missing trust anchor.
- We propose two deployment models of our solution and discuss the trade-offs in these models.
- We show the performance of our distributed RPKI system based on four threshold signature protocols, all of which have a stronger threat model than the existing RPKI system.
- We perform extensive evaluation of our system and show that our system is not only efficient for today's requirements, but it can also meet future demands.

Outline

Section 6.1 elaborates on the system and threat model of our system. Section 6.2 describes our distributed RPKI system. Section 6.3 discusses the performance of our distributed RPKI system. Section 6.4 analyses historical RPKI data to understand the number of ROAs issued/revoked over time and shows that our system satisfies the requirements. Section 6.5 discusses the related works.

6.1 System and Threat Model

In this section, we state the threat model of existing RPKI before introducing the system and communication model. We introduce different threat models from MPC literature that our solution supports along with the motivation behind each of them. Then, we introduce the system and communication model that we use in our work.

6.1.1 Threat Model

In our distributed RPKI system, we consider a stronger threat model than the existing RPKI system. The existing threat model of RPKI includes external adversaries, but not the participating entities, such as RIRs, to be a possible attacker. In this chapter, in addition to the threats considered in the existing system, we do not consider the RIRs to be entirely trustworthy.

Our threat model accounts for mistakes by the RIR as the hosted CA and the RIR under attack from an external adversary including legal coercion to modify, revoke or to inject RPKI data. All these scenarios require access to the signing key for the attack to work. We can capture these scenarios in our system by incorporating RIRs in the threat model. Note that attacks on the publication point, such as deletion of RPKI data, are beyond the scope of this chapter.

Standard MPC terminology provides us with a tool kit to discuss threat models that not only includes external adversaries but also the participating parties. We consider adversarial power, that is, whether an adversary is passive or active. Then, we describe the guarantees that can be achieved when the threshold of honest parties varies. Finally, we describe which guarantees our solution supports and how it translates to the threats against RPKI.

Semi-honest vs. malicious security. As we described in § 2.1, MPC protocols can be classified in terms of the power of the adversary. An adversary can be *semi-honest* or *malicious*. A semi-honest adversary follows the protocol while a malicious adversary does not follow the protocol and might actively disrupt the protocol. Security against semi-honest adversaries is sufficient in many real life scenarios. If the RIRs trust each other not to act maliciously and instead consider each other to be a necessary check on each other's operation, a protocol secure against a semi-honest adversary is sufficient. Such a protocol keeps the signing key away from any internal adversaries and curious employees at the RIRs.

However, semi-honest security is not sufficient when an adversary actively attempts to corrupt the computation. Such an adversary can maliciously send wrong values or delay the sending of values. A malicious adversary such as a nation-state actor can take full control over the RIRs and compromise the RIRs. They can attempt to inject wrongful information. Hence, we need to check the correctness of inputs to detect malicious activity. In contrast, a protocol secure against semi-honest adversaries assumes that the correct inputs are provided. The difference in efficiency depends on the specific instantiation of the protocols. We show in this chapter that protocols against malicious adversaries are efficient enough for the purpose of distributed RPKI.

Honest vs. dishonest majority. Our distribute RPKI system involves all the RIRs, that is, we use $n = 5$ for our threshold signature protocol. During the execution of a threshold signature protocol, a threshold t number of RIRs need to participate for the protocol to be successfully executed. When a majority of the RIRs are honest, then it is called *honest majority*. When a minority of the RIRs are honest, the protocol is said to be secure for a *dishonest majority*. In the case of honest majority protocols, as long as most of the RIRs do not collude, the key is not disclosed to anyone. As the RIRs often do not converge on the same policies, this may not be a strong assumption [MvEK11]. However, there are situations where a dishonest majority protocol might be needed as it provides stronger security such that the adversary needs to corrupt all the parties to be able to access the signing key. A dishonest majority protocol might also be required when the signature should only be created when there is unanimity among the RIRs.

6.1.2 System and Communication Model

Our system incorporates all the parts of the RPKI that requires generating signatures, which includes the creation of signed objects, ROAs, as well as signing of the resource certificates of children and the issuance of CRLs. Unlike traditional RPKI, we propose a Distributed RPKI (DRPKI) using MPC such that a ROA cannot be signed by individual RIRs on their own. Our system focuses on the role of RIRs as CAs in RPKI, specifically in the hosted RPKI setting where the RIR that allocates IP resources also runs the CA to validate the ROAs. We focus on the key generation and the signing operation in a distributed RPKI system, such that no RIR has access to the signing key. RIRs only have access to parts of the signing key (known as key shares) and not to the signing key. Thus, RIRs cannot unilaterally sign ROAs of ASes or revoke the associated end-entity certificate.

We inherit the communication model from the underlying MPC protocols. More specifically, we assume the existence of synchronous communication network where the protocol is executed in rounds [Can01, KMTZ13]. The communication between the RIRs is implemented and run on a point-to-point network. We assume that the RIRs securely communicate with each other using a TLS connection with authenticated endpoints.

6.2 Distributed RPKI

In this section, we describe the system setup in § 6.2.1. Then, we describe the different protocol phases of DRPKI in § 6.2.2. Finally, we discuss the deployment models for our system in § 6.2.3.

6.2.1 System Setup

We present the system architecture in this section. The setup at each RIR is shown in Figure 6.1 and the DRPKI architecture is shown in Figure 6.2. Each RIR has two components: trust anchor and hosted RPKI. Each RIR has a CA and a threshold signature module. *RIR CA* is the top-level CA that acts as a trust anchor in RPKI. RIR CA issues the CA certificates to its members and issues manifests and CRLs for the members. In addition, it also issues a self-signed certificate for itself and a certificate for the hosted CA. *Hosted CA* is responsible to produce signed objects: ROAs, CRL and manifests for the members. It is available to all members of the

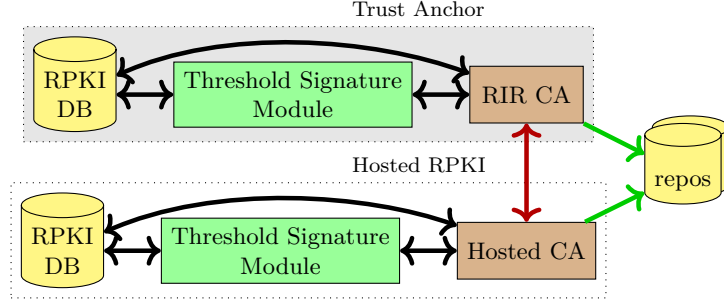


Figure 6.1: System setup.

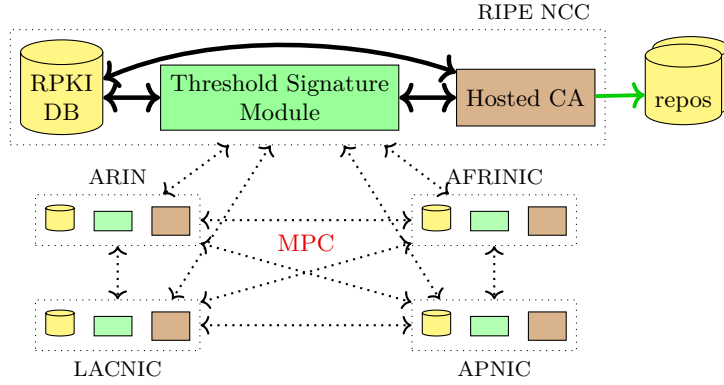


Figure 6.2: Distributed RPKI architecture.

RIR who choose to use hosted RPKI. The public key and the share of the private key of the members is stored at the hosted CA. All the certificates and the signed objects issued by the two CAs are published in public access repositories, through rsync or RPKI repository delta protocol [BMW17].

Unlike existing RPKI, our system requires interaction between the RIRs for the creation of certificates and signed objects (Figure 6.2). RIRs are in different continents and the communication takes place over the public internet using secure and authenticated channels. At a high-level, each RIR has a share of a private key for each member and uses this share to collaboratively issue signed objects. None of the RIRs get to access the entire private key. They use the shares of the private keys to create signed objects.

Key generation $([sk_j], pk_j) \leftarrow \text{KGen}(1^\lambda)$

1. Each RIR takes a security parameter 1^λ as the input and generates a signing key share for the j^{th} member by randomly sampling $[sk_j] \leftarrow \mathbb{Z}_p$.
2. Each RIR locally converts $[sk_j]$ to $[sk_j] \cdot G$.
3. RIRs compute the public key
 $pk_j = \text{Open}([sk_j] \cdot G) = sk_j \cdot G$.
4. Output the secret key shares and the public key $([sk_j], pk_j)$.

Figure 6.3: Key generation protocol.

6.2.2 DRPKI Protocol Phases

All phases are instigated by a CA and the interaction takes place between the threshold signing modules. MPC adds computation and communication overhead to traditional signing and, hence, we require a protocol that is efficient when the signing is to be performed. Protocols in the preprocessing model generate message independent material apriori and require little computation and communication to complete the signing when the message to be signed is available. Furthermore, most threshold signature protocols only satisfy some of the threat models we consider in our system. We want to be able to consider the efficiency of our system under all the threat models we discussed in § 6.1.1. Hence, we use the protocol described in § 5.3 as it is the most efficient protocol that fulfils all our requirements.

Key generation

In the key generation phase, new keys are generated such that each RIR generates a signing key share $[sk_j]$ for each member and runs the key generation protocol. At the end of this phase, each RIR has the public key pk_j and their share of the signing key $[sk_j]$. The key generation protocol needs to be run every time keys are to be generated. The keys do not need to be stored in a HSM. The complete signing key is not exposed unless a threshold number (depending on the protocol being honest majority or dishonest majority) of RIRs have been compromised or corrupt. Figure 6.3 describes the protocol details.

Signing

The threshold signing protocol we use has two preprocessing phases and one online phase. The first preprocessing phase is independent of the member for whom the signature is to be generated. More specifically, this phase is independent of the signing key to be used. This property allows us to amortize this phase. This phase can be run between the RIRs before the member's request to generate a signature arrives. Only an estimation of the number of signatures that would be required in a certain amount of time is required to run this phase. At the end of this phase, the desired number of initial preprocessing tuples are generated and stored at each RIR.

The second preprocessing phase is dependent on the member for whom the signature is to be generated. The threshold signature modules at the RIRs use one unused initial preprocessing tuple. It is security critical that the initial preprocessing tuples are not reused as it is equivalent to the reuse of the instance key k . An attacker with two messages signed using the same signing key and instance key can recover the signing key. At the end of this phase, the desired number of final preprocessing tuples are generated and stored at each RIR.

In the final signing phase, the member gives consent to the changes that can be made through a standalone application. This consent is sent to all the RIRs. When a signature is to be generated, the message to be signed is sent by the RIR that initiates the signing protocol to the other RIRs. The message can take the form of ROAs, CRLs, CA certificates or end-entity certificates. The message is checked, similar to the checks each RIR performs in the existing RPKI system, where they check the message locally before they individually sign. However, in our case, the check is performed by all the RIRs for all the messages that need to be signed. Furthermore, the consent of the member is checked. The RIRs check whether the consent has been given for the specific change, e.g., the transfer of IP space to another AS. Note that a transfer of IP-space requires consent for a CRL for the existing EE certificate associated with the ROA and to create a new signed ROA. These checks prevent RIRs to unilaterally take decisions to revoke certificates. If a threshold number of RIRs agree, then the RIRs locally compute their share of the signature before jointly computing the final signature.

Figure 6.4 describes the protocol details. With regard to the format of the messages, we do not make any change to the form and fields compared to the existing RPKI system. The certificates take the form of X.509 certificates [HML12]

Signing Protocol

Member Independent preprocessing from Figure 5.4.

$(\langle k \rangle, [k^{-1}]) \leftarrow \text{PInd}()$

Member Dependent preprocessing from Figure 5.4.

$(\langle k \rangle, [k^{-1}], [\text{sk}'_j]) \leftarrow \text{PDep}([\text{sk}_j], \langle k \rangle, [k^{-1}])$

Online signing. $\sigma \leftarrow \text{Sig}(\langle k \rangle, [k^{-1}], [\text{sk}'_j], M)$

1. The member uses a standalone application to give consent, e.g., to transfer IP-space to another AS. The consent is sent to all the RIRs.
2. Input the message to be signed M and the final preprocessed tuple $(\langle k \rangle, [k^{-1}], [\text{sk}'_j])$.
3. The RIR initiating the protocol sends the message M to the other RIRs.
4. The RIRs check the contents of M and the consent by the member before proceeding. If the check fails, they abort \perp . Else, they continue.
5. Then the RIRs compute
$$R \leftarrow \text{Open}(\langle k \rangle) = (bc^{-1}) \cdot G = a^{-1} \cdot G = k \cdot G.$$
6. Let $(r_x, r_y) \leftarrow R$.
7. Locally compute the share of the signature
$$[s] = H(M) \cdot [k^{-1}] + r_x \cdot [\text{sk}'_j].$$
8. Finally compute $s \leftarrow \text{Open}([s])$ and output the signature $\sigma = (r_x, s)$ or $\sigma = \perp$.

Figure 6.4: Signing protocol.

while the signed objects conform to RFC 6488 [LCK12]. The RIRs check the contents of the message out-of-band.

Automation

All the steps in our system are automated and they do not require human intervention at the RIRs. Note that Step 4 of online signing phase in Figure 6.4 requires checking the message before the message is signed. We automate this step through a simple consent mechanism during Step 1. For example, if the customer (AS1) of RIPE NCC is transferring an IP-space to AS2, then AS1 gives consent to revoke the old ROA and to issue the new ROA pointing to AS2. This consent is sent to all RIRs (instead of only to one RIR) from the customer facing software¹. Hence, all RIRs are informed of the intent of AS1. When the RIRs run the threshold signature protocol, there is an automatic check for consent. If a threshold number of RIRs have not received the consent, then the check fails and the automated signing protocol aborts.

Legitimate revocation without consent?

So far, we have assumed that revocation of allocated IP resources requires the consent of the INR holder. What about cases where there is a legitimate reason to revoke allocation? Let us take a case where ARIN was fraudulently induced to issue IPv4 addresses [ARI19]. After the fraud was detected and ARIN won a legal case, ARIN was able to take back the addresses. Using our automated system with enforced consent, revocation of the IP address space in such a scenario will not be possible. However, we are able to accommodate legitimate revocation with a minor change to the system.

Our automated system aborts the protocol if the check for consent fails at Step 4 of online signing phase in Figure 6.4. Instead of aborting the protocol, we can flag it with the requirement for manual intervention if the protocol is to be completed. Note that such a manual intervention will not require a large human effort as, in practice, most organizations obtain IP address space from their RIRs in good faith and there are only a few bad apples [RN19]. We will require the RIRs to communicate off band before the protocol is completed. This mechanism also allows for legitimate law enforcement requests to be processed by the RIRs, only when a threshold of

¹We need a standalone software to send the consent to all RIRs.

other RIRs also agree. Although technically possible, processing law enforcement mechanisms in this manner is akin to private regulation, which will require legal and policy changes for it to be realistic.

6.2.3 Deployment Scenarios

In this part, we propose different deployment models. We begin with a naïve deployment model and explain the reasons for its failure to solve the problem. Then, we present two solutions with their associated trade-offs among the stake holders. We emphasise that the trade-offs are not with respect to the security, but with respect to the responsibilities of the different stake holders.

Naïve solution

In a naïve solution, our threshold signing module can be used for hosted CAs. This solution allows for the existence of the delegated CAs, which are beneficial to ISPs who sub-allocate resources. This solution allows for INR holders to run their own CAs as well, that is, delegated CA system can continue to function in parallel with hosted CA system, as it does in the current system. So the ISPs which have their own CA can delegate INR holders and sign the ROAs. However, the only change is that the signing keys in the hosted CA setup are not in the possession of the individual RIRs. The trust anchor from the existing RPKI exists and the RIR CA that is higher in the hierarchy can still revoke certificates unilaterally as it is not distributed. And, the threat model remains weak and unchanged.

Our solutions

As the naïve deployment scenario does not solve the problem, we propose two deployment solutions. Both our solutions distribute the trust anchor (TA). Before discussing our solutions, we give an intuition behind our choice to distribute RPKI trust anchor. The notion of a TA requires all child nodes to unconditionally trust an entity. In RPKI, there are five TAs, one at each RIR, which the relying parties use to verify RPKI signatures. The concentration of power at TAs in the Internet infrastructure extends beyond RPKI and is also observable in DNS(SEC) and Web PKI. However, unlike DNS and Web PKI, there are already five TAs in RPKI that allows for a smooth transition to a distributed TA. Furthermore, the existing system

of five TAs has had its issues. As the policies of each RIR with regard to TA is different, some relying parties do not use the TA of ARIN and ROAs issued under ARIN’s trust anchor locator (TAL) fall to the status of ‘Not Found’ [Tin19]. This means that even when RPKI is implemented, a significant portion of the networks do not validate routes originating from North America due to policy decisions and legal barriers [CC19, YW19]. Thus, in practise, large parts of the world are prevented from having better routing security. These issues can be prevented if the TA is not located at individual RIRs with their own policies and is instead distributed across them.

Two-layered deployment. In our first solution, we propose a two-layered approach. The upper layer generates a distributed TA to the five RIRs, while the lower layer uses the threshold signing module for the hosted CAs. In both layers, the RIRs use our threshold signing module. In the upper layer, a distributed TA is established using our key generation protocol in Figure 6.3. Each RIR generates their signing key share and participates in the key generation protocol to obtain the public key. Once the public key is obtained, each RIR adds the public key to their TAL as the `subjectPublicKeyInfo` [CSF⁺08]. Each RIR has a TA that has the same public key in the TAL. As no RIR has the private key associated with this certificate, the RIR CAs do not need to be kept offline. Thus, the RIRs do not need a subordinate CA to issue child certificates. Furthermore, as each RIR has the same public key as part of the TA and they have the same `subjectPublicKeyInfo` in their TAL, access to the TAL from one RIR is sufficient for relying parties to validate routes originating from any part of the world that has deployed RPKI.

In the lower layer, our threshold signing module is used by the hosted CAs to generate signed objects such as ROAs. We are able to support delegated CAs as the distributed TA at the RIR CAs is used to generate child certificates. Furthermore, this solution allows for incremental deployment as the LIRs who have already deployed their own CAs can continue to use them to serve their child nodes while those who have not deployed their own CAs can start using hosted CA. Note that the concerns regarding some LIRs being coerced by their country of registration remains.

Flat deployment. In our second solution, instead of having the RIRs run two CAs, RIR CA and the hosted CA, we combine the two so that the RIRs only need

Majority	Adversary power	
	Semi-honest	Malicious
Honest	Shamir	Mal. Shamir
Dishonest	Semi. OT	MASCOT

Table 6.1: Four MPC protocols for distributed RPKI.

to run one CA. Furthermore, we do not need a TA as we replace the top-down architecture with a flat deployment architecture. Not only do we eliminate the hierarchical structure of existing RPKI, we also distribute trust. Moreover, this solution accounts for a stronger threat model where individual RIRs do not need to be completely trusted. However, we do not support delegated CAs in this solution. The CAs only generate end-entity certificates and signed objects; they do not generate any CA certificate that will allow child nodes to generate their own signed objects. This also means that child nodes will need the RIRs to generate signed objects for their child nodes. Nevertheless, we prevent any single entity to be all powerful and require the participation of a threshold number of RIRs for a signed object to be generated and ejected.

6.3 Implementation and Evaluation

We have implemented our system in C++ and have used MP-SPDZ [Dat19] for the threshold ECDSA protocols. MP-SPDZ includes threshold ECDSA protocol implementations for all the security models that we are concerned with: honest and dishonest majority protocols for honest-but-curious and malicious adversaries. In particular, we use four protocols—Shamir, Mal. Shamir, Semi. OT and MASCOT—that are shown in Table 6.1. The former two are based on Shamir secret sharing while the latter two are based on additive secret sharing. We discuss Shamir, Mal. Shamir and MASCOT in § 5.5.1. Semi. OT is similar to MASCOT but without any checks to detect malicious actions. We use it to show the performance of dishonest majority semi-honest protocols. We use all four protocols to implement the system described in § 6.2.2.

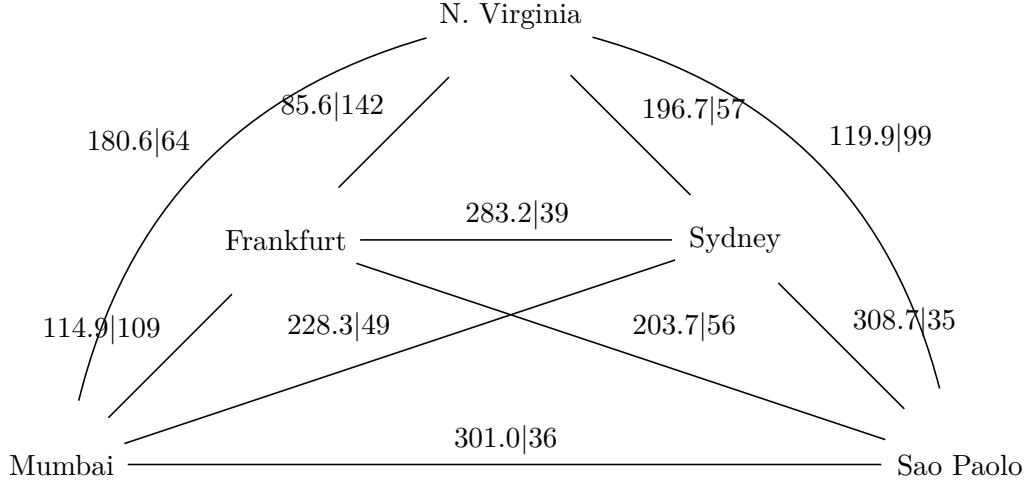


Figure 6.5: Latency|Bandwidth between regions, where latency is in milliseconds and bandwidth is in Mbits/s.

6.3.1 Deployment Setups

For performance evaluation, two deployments were set up. For each node, we used a VM instance AWS c5.2xlarge with a 64-bit Intel Xeon CPU with 3 GHz and 16 GB RAM. We run all the evaluations on a single thread. To make our evaluations as realistic as possible, we chose to run the experiments based on the location of the RIRs (See Table 6.2). The five RIRs are in different continents of the world. So, in the first setting, we run experiments on five AWS VMs that are placed around the world such that they are representative of the location of the RIRs. Specifically, we use the machines at Frankfurt, N.Virginia, Sydney, Sao Paolo and Mumbai while the RIRs are based in Amsterdam, Virginia, Brisbane, Sao Paolo, Mauritius, respectively. The latency and the bandwidth between the VMs are shown in Figure 6.5. Furthermore, we also consider the setting where the RIRs could, in the future, have virtual servers located close to other RIRs. For this purpose, we also run our experiments on the LAN in Frankfurt.

	Location	
	RIR	AWS
AFRINIC	Mauritius	Mumbai
APNIC	Brisbane	Sydney
ARIN	Virginia	N. Virginia
LACNIC	Sao Paolo	Sao Paolo
RIPE NCC	Amsterdam	Frankfurt

Table 6.2: Location of RIRs and AWS virtual machines.

	LAN			WAN		
	Secret	Public	KGen	Secret	Public	KGen
MASCOT	6.99 ± 0.04	2.48 ± 0.02	9.47 ± 0.03	4490 ± 1.74	1147 ± 0.27	5637 ± 1.25
Semi OT	0.88 ± 0.04	0.91 ± 0.02	1.79 ± 0.03	851 ± 2.53	486 ± 0.93	1337 ± 1.91
Mal. Shamir	0.24 ± 0.00	1.59 ± 0.03	1.83 ± 0.02	198 ± 0.23	487 ± 0.61	685 ± 0.46
Shamir	0.25 ± 0.04	1.13 ± 0.08	1.38 ± 0.06	284 ± 1.38	382 ± 3.48	666 ± 2.64

Table 6.3: Breakdown of key generation timings in milliseconds for $[sk]$ sharing, and pk .

6.3.2 Experimental Evaluations

Key generation. We benchmark the 5-party key generation protocol in both settings. The total key generation time is composed of the timings for generating secret key and public key. Secret key generation involves generating a field element $[sk]$ while public key generation involves a local conversion of the field element into an elliptic curve point of order p before being opened. The timings shown in Table 6.3 are the mean and standard deviation over 10 executions of the protocols where the value taken for each execution is the time noted when the last party completes the protocol. While the honest majority protocols (Shamir and Mal. Shamir) only require one round of communication for secret key generation, dishonest majority protocols (Semi OT and MASCOT) are costlier, especially in WAN setting.

Signing. We benchmark the preprocessing time (member dependent and independent) to generate tuples and the online signing time per signature in Table 6.4. For preprocessing, we present the time taken to generate one tuple when 1000 tuples are generated in an amortized manner. As the preprocessing does not depend on the message to be signed, thousands of preprocessed tuples can be generated and stored.

	LAN			WAN		
	Preprocessing	Online	Sig	Preprocessing	Online	Sig
MASCOT	4.78 ± 0.01	1.89 ± 0.02	6.67 ± 0.02	50.56 ± 1.86	1055 ± 37.23	1106 ± 26.36
Semi OT	0.96 ± 0.01	1.51 ± 0.01	2.47 ± 0.01	9.00 ± 0.90	487 ± 0.40	496 ± 0.70
Mal. Shamir	1.43 ± 0.00	1.40 ± 0.02	2.83 ± 0.01	10.94 ± 0.68	283 ± 0.06	294 ± 0.48
Shamir	0.98 ± 0.00	1.30 ± 0.02	2.28 ± 0.01	3.77 ± 0.00	282 ± 0.18	286 ± 0.13

Table 6.4: Breakdown of signing timings in milliseconds for preprocessing and online phases per signature. Preprocessing times are based on amortized generation of 1000 tuples.

They can be used when a new message is to be signed. Note that the online phase does not involve any elliptic curve operation and, hence, is computationally cheap.

Although dishonest majority protocols are generally costlier than honest majority protocols, Semi OT has the highest preprocessing throughput in LAN setting (Table 6.5). Semi OT protocol uses additive sharing that is cheaper than elliptic curve operations, which is the predominant cost during preprocessing. In the WAN setting, communication becomes more predominant than local operations. We also observe that the cost of malicious security in the case of honest majority protocol is very small. This is especially true in the WAN setting as the extra checks for Mal. Shamir are local operations and communication becomes the predominant cost.

In Table 6.6, we show the communication per party for the four protocols. We note that the communication is asymmetric for Mal. Shamir and Shamir. Hence, we present the mean of the communication over all the parties. We notice that the preprocessing communication per tuple as well as online signing is significantly higher for dishonest majority protocols than honest majority protocols. In comparison, the communication overhead per party is marginal for malicious security over honest-but-curious protocols.

6.4 Analysis

For the deployment of our distributed RPKI system, it needs to be efficient enough. In the previous section, we discussed the efficiency in terms of the runtime of our protocols. In this section, we discuss whether they are efficient enough in terms of the number of signatures required by RIRs. As our system involves all the five RIRs, we take into account the cumulative requirements of all of them.

	LAN		WAN	
	Preprocessing	Online	Preprocessing	Online
MASCOT	209	529	20	0.95
Semi OT	1042	662	111	2.05
Mal. Shamir	699	714	91	3.53
Shamir	1020	769	265	3.54

Table 6.5: Throughput for preprocessing (tuples/sec) and online phases (signatures/sec).

	KGen	Preprocessing (per tuple)	Online Signing
MASCOT	0.482	624	0.400
Semi OT	0.113	99.0	0.128
Mal. Shamir	0.271	1.345	0.0768
Shamir	0.206	0.437	0.0512

Table 6.6: Communication per party (KByte).

RPKI data. We accessed the publicly available historical RPKI data maintained by RIPE NCC that includes the daily archive of the repositories of all the five RIRs from 2011 onwards². We use the historical data from 11 March 2015 till 10 August 2020.

ROA analysis. We use RPKI data to analyse the number of ROAs that have been added and removed per day in a certain time period. We estimate the number of signatures required based on this information. Figure 6.6 shows the change on an average day (mean taken over a month) in ROAs for the five RIRs. On average, we need about 8000 signatures per day. However, there are days when the load is greater. This occurs on days when many ROAs are re-issued. Figure 6.7 shows the maximum number of changes per month. Note that the scale on y-axis is twenty-times that of Figure 6.6.

We observe from Table 6.5 that for our slowest protocol MASCOT, we are able to produce 0.95 signatures/sec or 82080 signatures/day in the WAN setting. For our fastest protocol, we are able to produce 3.54 signatures/sec or 305856 signatures/day in the WAN setting. Even our slowest protocol can produce 10x more signatures

²<https://ftp.ripe.net/rpki/>

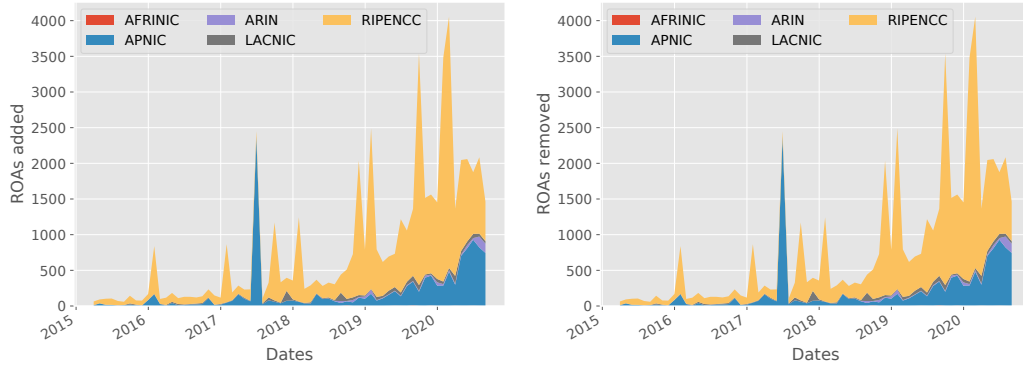


Figure 6.6: Average number of ROAs added and removed per day from March 2015 to August 2020.

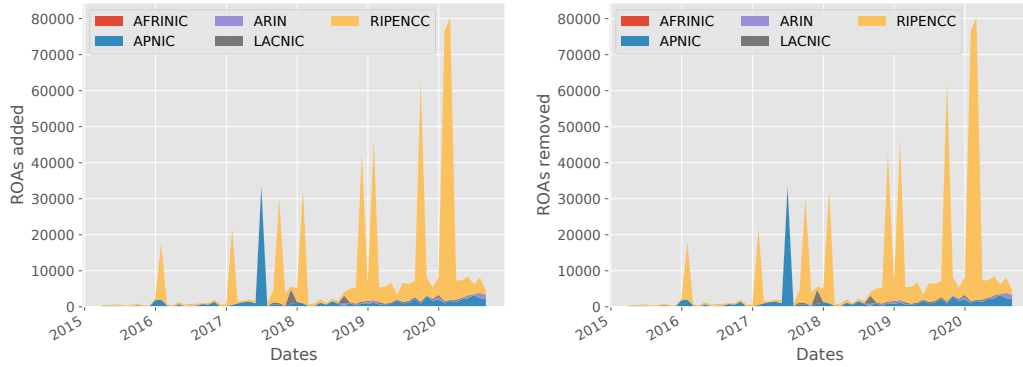


Figure 6.7: Maximum number of ROAs added and removed per month from March 2015 to August 2020.

than is required on an average day. All our other protocols are fast enough even on days with peaks in Figure 6.7. In the LAN setting, all our protocols are fast and have the capacity to produce three orders of magnitude more signatures. The efficiency of our system makes it possible to scale as the adoption of RPKI increases.

6.5 Related Works

One approach that has been proposed to address the issue of disproportionate power in the hands of RPKI authorities is to add transparency logs and `.dead` objects to RPKI to note the consent of the INR owner for revocation [HCRG14].

Heilman et al. [HCRG14] use a detector to identify when a ROA has downgraded from valid to invalid or valid to unknown state and check whether a `.dead` object is present. Three problems with their approach: (1) It requires effort not only from the CAs but also from relying parties. This method depends on relying parties to perform ROV and to alarm other relying parties. In practice, only about 100 ASes perform ROV as can be observed at ROV deployment monitor ³, a monitoring platform [RBC⁺18], while there are 68289 ASes as of 1 April 2020 ⁴. (2) It performs detection while we attempt to prevent malicious activities. Attacks can be detected after the fact due to the transparency and the effort of the relying parties. Accountability in the form of malicious activity detected post-mortem is not sufficient as ASes may have already lost out on their business. (3) These `.dead` objects are used to signify consent from the child and they are to be signed by the child node. In the hosted RPKI setting, as the parent manages signing for the child node, the parent can create and sign `.dead` objects by impersonating the child.

Another approach is to replace the existing RPKI system with blockchains [HL16]. This approach eliminates the possibility of RPKI authorities revoking previously allocated resource while they remain part of the blockchain by providing new resources. The use of blockchain raises other deployment issues such as consensus algorithm and incentive for the nodes to run the blockchain. If Proof-of-Stake is used as the consensus algorithm, as proposed in [PFG⁺18], then the nodes with greater stake, e.g., large ISPs who are allocated large subsets of IP addresses will become powerful players, which will create another form of power imbalance. As blockchain-based proposals suffer from scalability issues, RouteChain [SAA⁺19] employs a hierarchy of ASes which are assigned in subgroups to validate BGP announcements on the blockchain. However, in practice, ASes may have conflicting policies that prevent dynamic grouping and, thus, an incentive mechanism is also required.

³<https://rov.rpki.net>

⁴<https://www.caida.org/data/as-relationships/>

Summary and Future Work

There are more people with access to the Internet in 2020 than ever before. And yet, so many in the world do not have Internet access. All of us who use the Internet, knowingly or unknowingly, rely on the fundamental protocols that make the Internet infrastructure. While securing this infrastructure has taken the form of patch-work, it has also placed critical components of security, such as key management, in the hands of centralized authorities.

In this thesis, we ask the following question: *How can we improve the efficiency of generic MPC protocols that are secure against a diverse set of adversaries such that they can be used to secure the Internet infrastructure?* We answer this question in the affirmative by improving the practical efficiency of 2PC protocols and constructing efficient threshold signatures based on generic MPC before building systems that use these protocols to secure the Internet infrastructure without entirely relying on centralized authorities.

In Chapter 4, we demonstrate through extensive evaluations that the practical efficiency of 2PC based on garbled circuits can be improved by choosing transport layer protocols based on the network conditions and function to be computed using 2PC. The efficiency improvement through a better use of transport layer protocols had not been explored before. We develop a modular framework, Transputation that includes multiple transport layer protocols and 2PC implementations. a direction that has not been explored before.

In Chapter 5, we construct efficient threshold signature protocols based on generic MPC that are secure in a diverse set of adversarial models. We transform any MPC protocol into an equally secure and efficient protocol that computes threshold ECDSA

signatures. Our transformation may have other applications. Our protocols are in the preprocessing model, wherein most of the cryptographic computation is performed in advance. A fast online phase in addition to the preprocessing results in protocols that are efficient and can produce hundreds of signatures per second.

Then, we use our threshold signature protocols to construct and implement a multiparty zone signing system that secures DNSSEC keys. Our system prevents individual DNS operators from accessing the signing keys while not hindering its ability function and generate DNSSEC signatures. Furthermore, when using threshold signature protocols secure against malicious adversaries, our system is secure even when the operational integrity of the operators is compromised, e.g., when an operator is coerced by nation-state actors. We also assess the applicability of our system through a measurement study that estimates the number of domains which already use multiple operators.

In Chapter 6, we observe that the existing design of RPKI makes IP prefix takedown a possibility as it has a weak threat model with centralized authorities in the form of RIRs. We strengthen the threat model and construct a distributed RPKI system that relies on threshold signatures. Our system prevents rather than detects IP prefix takedown. We also distribute the trust anchor to prevent situations where RPKI validation fails due to missing trust anchor. Not only is our system automated to prevent manual configuration errors, it also does not require any changes at the relying parties, making it easier to deploy. We evaluate the efficiency of our system and show that it can handle the present-day load of RPKI as well as scale for future increase in RPKI deployment.

Through the construction of efficient MPC protocols and their use in building systems, we have been able to solve specific problems of centralization with respect to key management in DNSSEC and RPKI. We have proposed protocol techniques, and designed and evaluated our systems. Nonetheless, there are many more related problems that are yet to be resolved. This thesis opens opportunities for further research, with which we conclude this thesis.

Future Work

There are research directions that are yet to be explored. We provide pointers for further exploration to conclude this thesis.

We study 2PC based on garbled circuits with *semi-honest* security. We conjecture that our results would naturally extend to protocols with *malicious* security: this conjecture is based on the observation that state-of-the-art protocols with active security are based on a technique known as *cut-and-choose* in which multiple copies of garbled circuits are sent, checked and evaluated [LR15, KNR⁺17]. In particular, the main difference between these protocols and passively secure protocols is not in the type of operation performed, but mostly in the higher-bandwidth requirements. Due to this replication factor we conjecture that, in the active security setting, SABUL will start outperforming TCP at smaller circuit size.

Furthermore, studying the impact of transport layer protocols in the multiparty case would also be interesting. There are many different protocols based on the corruption threshold (e.g., honest majority vs. dishonest majority), the number of parties, the protocol design and the communication pattern. Different MPC implementations exhibit distinct communication patterns (e.g., communication rounds and volume of data transmitted during each round). Future work is needed to devise specially engineered transport protocols for specific computation tasks and applications. This is an important yet non-trivial research direction, and requires investigation of the specific properties of the target application as well as the network conditions in which they are run.

The threshold signature protocols we design and use in the systems we build are in the security-with-abort model. Malicious parties can abort the computation and prevent honest parties from getting the output. In some cases, we might need accountability so that we can attribute which party aborted the protocol. In the case of our honest-majority protocols, we can construct the signatures as long as a threshold number of parties remain honest. If some party sends incorrect inputs, then honest parties can identify this corrupt party through additional local computation. However, we might want a stronger guaranty so that the party that aborted can be identifiable by an external party, e.g., for the purpose of audit. In such cases, we might extend our protocols to security-with-identifiable-abort model. While Gennaro and Goldfeder [GG20] propose a tailored dishonest majority protocol in security-with-identifiable-abort model, it will be interesting if we can

extend threshold signatures based on generic MPC protocols in a black-box manner.

Extending our protocols to obtain stronger guarantees such as fairness (either everyone receives a signature, or no one does) and guaranteed output delivery (a signature is guaranteed to be output, regardless of the actions of corrupt parties) is also an interesting direction for research. While these extensions for generic MPC are theoretically possible [CHOR18], practically efficient transformations that can exploit the properties of signatures would be beneficial. In some applications, the availability of the system might be of the utmost importance. Any form of denial of service may not be acceptable. For instance, some domains cannot afford to have any down time. In such cases, our protocols should have guaranteed output delivery.

The multizone signing system and the distributed RPKI system that we design are purely technical solutions. For them to be deployed, many non-technical issues need to be resolved. There are policy and legal barriers that need to be addressed. Our systems require cross-border communication. In the case of the distributed RPKI system, the internal policies of each RIR need to be changed as well as the policies between the RIRs need to be reconsidered for our system to be deployed.

Bibliography

- [AA04] Derek Atkins and Rob Austein. Threat analysis of the domain name system (DNS). RFC 3833, RFC Editor, August 2004.
- [AAL⁺05a] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose. DNS security introduction and requirements. RFC 4033, RFC Editor, March 2005.
- [AAL⁺05b] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose. Protocol modifications for the DNS security extensions. RFC 4035, RFC Editor, March 2005.
- [AAL⁺05c] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose. Resource records for the DNS security extensions. RFC 4034, RFC Editor, March 2005.
- [ABL⁺18] David W. Archer, Dan Bogdanov, Yehuda Lindell, Liina Kamm, Kurt Nielsen, Jakob Illeborg Pagter, Nigel P. Smart, and Rebecca N. Wright. From keys to databases - real-world applications of secure multi-party computation. *Comput. J.*, 61(12):1749–1771, 2018.
- [AIK14] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. *SIAM J. Comput.*, 43(2):905–929, 2014.
- [Ale19] Alexa. Top 1m sites, July 12 2019. <https://www.alexa.com/topsites>.
- [ALSZ13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 535–548. ACM, 2013.
- [AM07] Suranjith Ariyapperuma and Chris J. Mitchell. Security vulnerabilities in DNS and DNSSEC. In *Proceedings of the The Second*

- International Conference on Availability, Reliability and Security, ARES 2007, The International Dependability Conference - Bridging Theory and Practice, April 10-13 2007, Vienna, Austria*, pages 335–342. IEEE Computer Society, 2007.
- [ARI19] ARIN. ARIN wins important legal case and precedent against fraud, 13 May 2019. https://www.arin.net/vault/about_us/media/releases/20190513.html Accessed: 30 November 2020.
- [AvRN18] Abhishta Abhishta, Roland van Rijswijk-Deij, and Lambert J. M. Nieuwenhuis. Measuring the impact of a successful DDoS attack on the customer behaviour of managed DNS service providers. *Computer Communication Review*, 48(5):70–76, 2018.
- [BB89] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In Piotr Rudnicki, editor, *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing, Edmonton, Alberta, Canada, August 14-16, 1989*, pages 201–209. ACM, 1989.
- [BCD⁺09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009.
- [Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991.
- [Bea95] Donald Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109. Springer, 1995.
- [Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In Gary L. Miller, editor, *Proceedings*

- of the *Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 479–488. ACM, 1996.
- [Bee16] Kris Beavers. A note from NS1’s CEO: How we responded to last week’s major, multi-faceted DDoS attacks, May 23, 2016. <https://ns1.com/blog/how-we-responded-to-last-weeks-major-multi-faceted-ddos-attacks> Accessed: 30 November 2020.
- [Bel95] Steven M. Bellovin. Using the domain name system for system break-ins. In Frederick M. Avolio and Steven M. Bellovin, editors, *Proceedings of the 5th USENIX Security Symposium, Salt Lake City, Utah, USA, June 5-7, 1995*. USENIX Association, 1995.
- [BFMR10] Kevin R. B. Butler, Toni R. Farley, Patrick D. McDaniel, and Jennifer Rexford. A survey of BGP security issues and solutions. *Proceedings of the IEEE*, 98(1):100–122, 2010.
- [BFZ07] Hitesh Ballani, Paul Francis, and Xinyang Zhang. A study of prefix hijacking and interception in the internet. In Jun Murai and Kenjiro Cho, editors, *Proceedings of the ACM SIGCOMM 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, August 27-31, 2007*, pages 265–276. ACM, 2007.
- [BGG17] Dan Boneh, Rosario Gennaro, and Steven Goldfeder. Using level-1 homomorphic encryption to improve threshold DSA signatures for bitcoin wallet security. In Tanja Lange and Orr Dunkelman, editors, *Progress in Cryptology - LATINCRYPT 2017 - 5th International Conference on Cryptology and Information Security in Latin America, Havana, Cuba, September 20-22, 2017, Revised Selected Papers*, volume 11368 of *Lecture Notes in Computer Science*, pages 352–377. Springer, 2017.
- [BHKL18] Assi Barak, Martin Hirt, Lior Koskas, and Yehuda Lindell. An end-to-end system for large scale P2P MPC-as-a-service and low-bandwidth MPC for weak participants. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 695–712. ACM, 2018.
- [BHKR13] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *2013*

- IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 478–492. IEEE Computer Society, 2013.
- [BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 784–796. ACM, 2012.
- [BJSV15] Dan Bogdanov, Marko Jöemets, Sander Siim, and Meril Vaht. How the estonian tax and customs board evaluated a tax fraud detection system based on secure multi-party computation. In Rainer Böhme and Tatsuaki Okamoto, editors, *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, volume 8975 of *Lecture Notes in Computer Science*, pages 227–234. Springer, 2015.
- [BL88] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer, 1988.
- [BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513. ACM, 1990.
- [BMWA17] Tim Bruijnzeels, Oleg Muravskiy, Bryan Weber, and Rob Austein. The RPKI repository delta protocol (RRDP). RFC 8182, RFC Editor, July 2017.
- [Bor15] Stephane Bortzmeyer. DNS privacy considerations. RFC 7626, RFC Editor, August 2015.
- [Bor16] Stephane Bortzmeyer. DNS query name minimisation to improve privacy. RFC 7816, RFC Editor, March 2016.
- [Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001*,

- Las Vegas, Nevada, USA*, pages 136–145. IEEE Computer Society, 2001.
- [CC19] Ben Cartwright-Cox. The State of RPKI: Q4 2018, 20 December 2019. <https://blog.benjojo.co.uk/post/state-of-rpki-in-2018> Accessed: 30 November 2020.
- [CCG⁺16] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. BBR: congestion-based congestion control. *ACM Queue*, 14(5):20–53, 2016.
- [CDN15] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.
- [CF04] Carlo Caini and Rosario Firrincieli. TCP hybla: a TCP enhancement for heterogeneous networks. *Int. J. Satellite Communications Networking*, 22(5):547–566, 2004.
- [CG09] Sergio Castillo-Perez and Joaquín García-Alfaro. Evaluation of two privacy-preserving protocols for the DNS. In Shahram Latifi, editor, *Sixth International Conference on Information Technology: New Generations, ITNG 2009, Las Vegas, Nevada, USA, 27-29 April 2009*, pages 411–416. IEEE Computer Society, 2009.
- [CHB⁺13] Danny Cooper, Ethan Heilman, Kyle Brogle, Leonid Reyzin, and Sharon Goldberg. On the risk of misbehaving RPKI authorities. In Dave Levine, Sachin Katti, and Dave Oran, editors, *Twelfth ACM Workshop on Hot Topics in Networks, HotNets-XII, College Park, MD, USA, November 21-22, 2013*, pages 16:1–16:7. ACM, 2013.
- [CHM⁺16] Francisco Cifuentes, Alejandro Hevia, Francisco Montoto, Tomás Barros, Victor Ramiro, and Javier Bustos-Jiménez. Poor man’s hardware security module (pmHSM): A threshold cryptographic backend for DNSSEC. In *Proceedings of the 9th Latin America Networking Conference, LANC 2016, Valparaiso, Chile, October 13-14, 2016*, pages 59–64. ACM, 2016.
- [CHOR18] Ran Cohen, Iftach Haitner, Eran Omri, and Lior Rotem. From fairness to full security in multiparty computation. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, volume 11035 of *Lecture Notes in Computer Science*, pages 216–234. Springer, 2018.

- [Cim19] Catalin Cimpanu. Hackers breached Greece’s top-level domain registrar, July 9, 2019. <https://www.zdnet.com/article/hackers-breached-greeces-top-level-domain-registrar/> Accessed: 30 November 2020.
- [CKKZ12] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the security of the “free-xor” technique. In Ronald Cramer, editor, *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 39–53. Springer, 2012.
- [CMP11] Emiliano De Cristofaro, Mark Manulis, and Bertram Poettering. Private discovery of common social contacts. In Javier López and Gene Tsudik, editors, *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*, volume 6715 of *Lecture Notes in Computer Science*, pages 147–165, 2011.
- [CMP20] Ran Canetti, Nikolaos Makriyannis, and Udi Peled. UC Non-Interactive, Proactive, Threshold ECDSA. Cryptology ePrint Archive, Report 2020/492, 2020. <https://eprint.iacr.org/2020/492>.
- [Cow10] Jim Cowie. China’s 18-minute mystery, 18 November 2010. <https://web.archive.org/web/20200109211935/https://dyn.com/blog/chinas-18-minute-mystery/> Accessed: 30 November 2020.
- [CS04] Christian Cachin and Asad Samar. Secure distributed DNS. In *2004 International Conference on Dependable Systems and Networks (DSN 2004), 28 June - 1 July 2004, Florence, Italy, Proceedings*, pages 423–432. IEEE Computer Society, 2004.
- [CSF⁺08] David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, and W. Timothy Polk. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. RFC 5280, RFC Editor, May 2008.
- [CvRC⁺17] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David R. Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. A longitudinal, end-to-end view of the DNSSEC ecosystem. In Engin Kirda and Thomas Ristenpart, editors, *26th USENIX Security Symposium, USENIX Security 2017*,

- Vancouver, BC, Canada, August 16-18, 2017, pages 1307–1322. USENIX Association, 2017.
- [Dat19] Data61. MP-SPDZ - versatile framework for multi-party computation, June 7, 2019. Version 0.1.0. <https://github.com/data61/MP-SPDZ>.
- [Des87] Yvo Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 120–127. Springer, 1987.
- [DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, 1989.
- [DGR18] Sara Dickinson, Daniel Kahn Gillmor, and Tirumaleswar Reddy. Usage profiles for DNS over TLS and DNS over DTLS. RFC 8310, RFC Editor, March 2018.
- [DJN⁺20] Ivan Damgård, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Jakob Illeborg Pagter, and Michael Bæksvang Østergaard. Fast threshold ECDSA with honest majority. In Clemente Galdi and Vladimir Kolesnikov, editors, *Security and Cryptography for Networks - 12th International Conference, SCN 2020, Amalfi, Italy, September 14-16, 2020, Proceedings*, volume 12238 of *Lecture Notes in Computer Science*, pages 382–400. Springer, 2020.
- [DK01] Ivan Damgård and Maciej Koprowski. Practical threshold RSA signatures without a trusted dealer. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 152–165. Springer, 2001.
- [DKL⁺12] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority – or: Breaking the SPDZ limits. Cryptology ePrint Archive, Report 2012/642, 2012. <https://eprint.iacr.org/2012/642>.

- [DKLS18] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 980–997. IEEE Computer Society, 2018.
- [DKLS19] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1051–1066. IEEE, 2019.
- [DLZ⁺15] Mo Dong, Qingxi Li, Doron Zarchy, Philip Brighten Godfrey, and Michael Schapira. PCC: re-architecting congestion control for consistent high performance. In *12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 15, Oakland, CA, USA, May 4-6, 2015*, pages 395–408. USENIX Association, 2015.
- [DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.
- [DSZ15] Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*. The Internet Society, 2015. Code: <https://github.com/encryptogroup/ABY>.
- [DYN08] DYN. Pakistan hijacks youtube, 24 February 2008. <https://web.archive.org/web/20200512071159/https://dyn.com/blog/pakistan-hijacks-youtube-1/> Accessed: 30 November 2020.
- [DYN16] DYN. Dyn analysis summary of friday october 21 attack, October 26, 2016. <https://web.archive.org/web/20200625183317/https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/> Accessed: 30 November 2020.

- [EBBP97] R. Elz, R. Bush, S. Bradner, and M. Patton. Selection and operation of secondary dns servers. BCP 16, RFC Editor, July 1997.
- [EFL12] Yael Ejgenberg, Moriya Farbstain, Meital Levy, and Yehuda Lindell. SCAPI: the secure computation application programming interface. *IACR Cryptology ePrint Archive*, 2012:629, 2012. Code: <https://github.com/cryptobiu/libscapi>.
- [FLNW17] Jun Furukawa, Yehuda Lindell, Ariel Nof, and Or Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 225–255, 2017.
- [FT14] S. Farrell and H. Tschofenig. Pervasive monitoring is an attack. BCP 188, RFC Editor, May 2014. <http://www.rfc-editor.org/rfc/rfc7258.txt>.
- [GCH⁺17] Yossi Gilad, Avichai Cohen, Amir Herzberg, Michael Schapira, and Haya Shulman. Are We There Yet? On RPKI’s Deployment and Security. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society, 2017.
- [GG03] Yunhong Gu and Robert L. Grossman. SABUL: A transport protocol for grid computing. *J. Grid Comput.*, 1(4):377–386, 2003.
- [GG07] Yunhong Gu and Robert L. Grossman. UDT: UDP-based data transfer for high-speed wide area networks. *Computer Networks*, 51(7):1777–1799, 2007.
- [GG08] Yunhong Gu and Robert L. Grossman. UDTv4: improvements in performance and usability. In Pascale Vicat-Blanc Primet, Tomohiro Kudoh, and Joe Mambretti, editors, *Networks for Grid Applications, Second International Conference, GridNets 2008, Beijing, China, October 8-10, 2008, Revised Selected Papers*, volume 2 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 9–23. Springer, 2008.

- [GG18] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In David Lie, Mohammad Manan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1179–1194. ACM, 2018.
- [GG20] Rosario Gennaro and Steven Goldfeder. One round threshold ecdsa with identifiable abort. Cryptology ePrint Archive, Report 2020/540, 2020. <https://eprint.iacr.org/2020/540>.
- [GGN16] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve A. Schneider, editors, *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, volume 9696 of *Lecture Notes in Computer Science*, pages 156–174. Springer, 2016.
- [Gil99] Niv Gilboa. Two party RSA key generation. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 116–129. Springer, 1999.
- [GJKR96] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 354–371. Springer, 1996.
- [GKP⁺13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nikolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 555–564. ACM, 2013.
- [GLNP15] Shay Gueron, Yehuda Lindell, Ariel Nof, and Benny Pinkas. Fast garbling of circuits under standard assumptions. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications*

- Security, Denver, CO, USA, October 12-16, 2015*, pages 567–578. ACM, 2015.
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.
- [Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004.
- [HAD⁺20] S. Huque, P. Aras, J. Dickinson, J. Vcelak, and D. Blacka. Multi-signer DNSSEC models. RFC 8901, RFC Editor, September 2020.
- [Hal18] Shai Halevi. Advanced cryptography: Promise and challenges. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, page 647. ACM, 2018.
- [HAN02] Thomas J. Hacker, Brian D. Athey, and Brian Noble. The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network. In *16th International Parallel and Distributed Processing Symposium (IPDPS 2002), 15-19 April 2002, Fort Lauderdale, FL, USA, CD-ROM/Abstracts Proceedings*. IEEE Computer Society, 2002.
- [HBZ17] Mario Hock, Roland Bless, and Martina Zitterbart. Experimental evaluation of BBR congestion control. In *25th IEEE International Conference on Network Protocols, ICNP 2017, Toronto, ON, Canada, October 10-13, 2017*, pages 1–10. IEEE Computer Society, 2017.
- [HCRG14] Ethan Heilman, Danny Cooper, Leonid Reyzin, and Sharon Goldberg. From the consent of the routed: improving the transparency of the RPKI. In Fabián E. Bustamante, Y. Charlie Hu, Arvind Krishnamurthy, and Sylvia Ratnasamy, editors, *ACM SIGCOMM 2014 Conference, SIGCOMM’14, Chicago, IL, USA, August 17-22, 2014*, pages 51–62. ACM, 2014.
- [HEK12] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *19th Annual Network and Distributed System Security Symposium,*

- NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012.
- [HEKM11] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*. USENIX Association, 2011.
- [HHNZ19] Marcella Hastings, Brett Hemenway, Daniel Noble, and Steve Zdancewic. SoK: general purpose compilers for secure multi-party computation. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1220–1237. IEEE, 2019.
- [HHSW18] Tomas Hlavacek, Amir Herzberg, Haya Shulman, and Michael Waidner. Practical experience: Methodologies for measuring route origin validation. In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2018, Luxembourg City, Luxembourg, June 25-28, 2018*, pages 634–641. IEEE Computer Society, 2018.
- [HL16] Adishesu Hari and T. V. Lakshman. The internet blockchain: A distributed, tamper-resistant transaction framework for the internet. In Bryan Ford, Alex C. Snoeren, and Ellen W. Zegura, editors, *Proceedings of the 15th ACM Workshop on Hot Topics in Networks, HotNets 2016, Atlanta, GA, USA, November 9-10, 2016*, pages 204–210. ACM, 2016.
- [HLOI16] Brett Hemenway, Steve Lu, Rafail Ostrovsky, and William Welser IV. High-precision secure computation of satellite collision probabilities. In Vassilis Zikas and Roberto De Prisco, editors, *Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy, August 31 - September 2, 2016, Proceedings*, volume 9841 of *Lecture Notes in Computer Science*, pages 169–187. Springer, 2016.
- [HM18] Paul E. Hoffman and Patrick McManus. DNS queries over HTTPS (DoH). RFC 8484, RFC Editor, October 2018.
- [HMF14] Dominik Herrmann, Max Maaß, and Hannes Federrath. Evaluating the security of a DNS query obfuscation scheme for private web surfing. In Nora Cuppens-Boulahia, Frédéric Cuppens, Sushil Jajodia, Anas Abou El Kalam, and Thierry Sans, editors, *ICT*

- Systems Security and Privacy Protection - 29th IFIP TC 11 International Conference, SEC 2014, Marrakech, Morocco, June 2-4, 2014. Proceedings*, volume 428 of *IFIP Advances in Information and Communication Technology*, pages 205–219. Springer, 2014.
- [HML12] Geoff Huston, George Michaelson, and Robert Loomans. A profile for X.509 PKIX resource certificates. RFC 6487, RFC Editor, February 2012.
- [HRA11] Geoff Huston, Mattia Rossi, and Grenville J. Armitage. Securing BGP - A literature survey. *IEEE Commun. Surv. Tutorials*, 13(2):199–222, 2011.
- [HRX08] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: a new tcp-friendly high-speed TCP variant. *ACM SIGOPS Oper. Syst. Rev.*, 42(5):64–74, 2008.
- [HS13a] Amir Herzberg and Haya Shulman. Fragmentation considered poisonous, or: One-domain-to-rule-them-all.org. In *IEEE Conference on Communications and Network Security, CNS 2013, National Harbor, MD, USA, October 14-16, 2013*, pages 224–232. IEEE, 2013.
- [HS13b] Amir Herzberg and Haya Shulman. Socket overloading for fun and cache-poisoning. In Charles N. Payne Jr., editor, *Annual Computer Security Applications Conference, ACSAC '13, New Orleans, LA, USA, December 9-13, 2013*, pages 189–198. ACM, 2013.
- [HW12] Paul E. Hoffman and Wouter C. A. Wijngaards. Elliptic curve digital signature algorithm (DSA) for DNSSEC. RFC 6605, RFC Editor, April 2012.
- [HWM⁺19] Geoff Huston, Samuel Weiler, George Michaelson, Stephen T. Kent, and Tim Bruijnzeels. Resource public key infrastructure (RPKI) trust anchor locator. RFC 8630, RFC Editor, August 2019.
- [ICA14] ICANN. ICANN Tells U.S. Court That ccTLDs Are Not “Property” | Files Motion to Quash in U.S. Legal Action Aimed at Seizing Top-Level Domains, 30 July 2014. <https://www.icann.org/resources/press-material/release-2014-07-30-en> Accessed: 30 November 2020.
- [IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology*

- Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003.
- [IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 44–61. ACM, 1989.
- [Jan09] Jelte Jansen. Use of SHA-2 algorithms with RSA in DNSKEY and RRSIG resource records for DNSSEC. RFC 5702, RFC Editor, October 2009.
- [Kam08] Dan Kaminsky. Black ops 2008: It’s the end of the cache as we know it. *Black Hat USA*, 2008.
- [KG13] C Kerry and P Gallagher. FIPS PUB 186-4: Digital Signature Standard (DSS). *Federal Information Processing Standards Publication. National Institute of Standards and Technology*, 2013.
- [KM17] S. Kent and D. Ma. Adverse Actions by a Certification Authority (CA) or Repository Manager in the Resource Public Key Infrastructure (RPKI). RFC 8211, RFC Editor, September 2017.
- [KMG12] O. Kolkman, W. Mekking, and R. Gieben. Dnssec operational practices, version 2. RFC 6781, RFC Editor, December 2012.
- [KMTZ13] Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In Amit Sahai, editor, *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, volume 7785 of *Lecture Notes in Computer Science*, pages 477–498. Springer, 2013.
- [Kno19] Knot. Knot DNS, July 1, 2019. Version 2.8. <https://www.knot-dns.cz/>.
- [KNR⁺17] Vladimir Kolesnikov, Jesper Buus Nielsen, Mike Rosulek, Ni Trieu, and Roberto Trifiletti. DUPLO: unifying cut-and-choose for garbled circuits. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 3–20. ACM, 2017.

- [KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 830–842. ACM, 2016.
- [Kre17] Ben Kreuter. Secure multiparty computation at google. In *Real World Crypto Conference (RWC)*, 2017.
- [Kre19] Krebs on Security. A deep dive on the recent widespread DNS hijacking attacks, February 18, 2019. <https://krebsonsecurity.com/2019/02/a-deep-dive-on-the-recent-widespread-dns-hijacking-attacks/> Accessed: 30 November 2020.
- [KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfssdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings*, volume 5126 of *Lecture Notes in Computer Science*, pages 486–498. Springer, 2008.
- [KW15] Liina Kamm and Jan Willemson. Secure floating point arithmetic and private satellite collision analysis. *Int. J. Inf. Sec.*, 14(6):531–548, 2015.
- [KZe19] KZen networks. Multi-party ECDSA, 2019. Version 0.2.5. <https://github.com/KZen-networks/multi-party-ecdsa>.
- [LBS08] Shao Liu, Tamer Basar, and R. Srikant. TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks. *Perform. Eval.*, 65(6-7):417–440, 2008.
- [LCK12] Matt Lepinski, Andrew Chi, and Stephen T. Kent. Signed object template for the resource public key infrastructure (RPKI). RFC 6488, RFC Editor, February 2012.
- [LFS⁺15] Suqi Liu, Ian D. Foster, Stefan Savage, Geoffrey M. Voelker, and Lawrence K. Saul. Who is .com?: Learning to parse WHOIS records. In Kenjiro Cho, Kensuke Fukuda, Vivek S. Pai, and Neil Spring, editors, *Proceedings of the 2015 ACM Internet Measurement*

- Conference, IMC 2015, Tokyo, Japan, October 28-30, 2015*, pages 369–380. ACM, 2015.
- [Lin17] Yehuda Lindell. Fast secure two-party ECDSA signing. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 613–644. Springer, 2017.
- [LK12] Matt Lepinski and Stephen T. Kent. An infrastructure to support secure internet routing. RFC 6480, RFC Editor, February 2012.
- [LKS04] Charles Lynn, Stephen T. Kent, and Karen Seo. X.509 extensions for IP addresses and AS identifiers. RFC 3779, RFC Editor, June 2004.
- [LNR18] Yehuda Lindell, Ariel Nof, and Samuel Ranellucci. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. Cryptology ePrint Archive, Report 2018/987, 2018. <https://eprint.iacr.org/2018/987>.
- [LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*, pages 52–78. Springer, 2007.
- [LP09] Yehuda Lindell and Benny Pinkas. A proof of security of yao’s protocol for two-party computation. *J. Cryptology*, 22(2):161–188, 2009.
- [LR15] Yehuda Lindell and Ben Riva. Blazing fast 2PC in the offline/online setting with security for malicious adversaries. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 579–590. ACM, 2015.
- [LS17] Matt Lepinski and Kotikalapudi Sriram. BGPsec protocol specification. RFC 8205, RFC Editor, September 2017.

- [LWN⁺15] Chang Liu, Xiao Shaun Wang, Kartik Nayak, Yan Huang, and Elaine Shi. OblivM: A programming framework for secure computation. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 359–376. IEEE Computer Society, 2015.
- [MNPS04] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay - secure two-party computation system. In Matt Blaze, editor, *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 287–302. USENIX, 2004.
- [Moc87a] Paul V. Mockapetris. Domain names - concepts and facilities. STD 13, RFC Editor, November 1987.
- [Moc87b] Paul V. Mockapetris. Domain names - implementation and specification. STD 13, RFC Editor, November 1987.
- [MR01] Philip D. MacKenzie and Michael K. Reiter. Two-party generation of DSA signatures. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 137–154. Springer, 2001.
- [MvEK11] M. Mueller, M. van Eeten, and B. Kuerbis. In important case, ripe-ncc seeks legal clarity on how it responds to foreign court orders, 23 November 2011. <https://www.internetgovernance.org/2011/11/23/in-important-case-ripe-ncc-seeks-legal-clarity-on-how-it-responds-to-foreign-court-orders/> Accessed: 30 November 2020.
- [Nag84] John Nagle. Congestion control in IP/TCP internetworks. RFC 896, RFC Editor, January 1984.
- [Net19] Netnod. Statement on man-in-the-middle attack against netnod, February 5, 2019. <https://www.netnod.se/news/statement-on-man-in-the-middle-attack-against-netnod> Accessed: 30 November 2020.
- [NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, pages 448–457. ACM/SIAM, 2001.

- [NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In Stuart I. Feldman and Michael P. Wellman, editors, *Proceedings of the First ACM Conference on Electronic Commerce (EC-99), Denver, CO, USA, November 3-5, 1999*, pages 129–139. ACM, 1999.
- [NST17] Jesper Buus Nielsen, Thomas Schneider, and Roberto Trifiletti. Constant round maliciously secure 2PC with function-independent preprocessing using LEGO. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society, 2017.
- [Ped91] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526. Springer, 1991.
- [PFG⁺18] Jordi Paillisse, Miquel Ferriol, Eric Garcia, Hamid Latif, Carlos Piris, Albert Lopez-Bresco, Brenden Kuerbis, Alberto Rodríguez-Natal, Vina Ermagan, Fabio Maino, and Albert Cabellos. IPchain: securing IP prefix allocation and delegation with blockchain. In *IEEE International Conference on iThings/GreenCom/CPSCoM/SmartData 2018, Halifax, NS, Canada, July 30 - August 3, 2018*, pages 1236–1243. IEEE, 2018.
- [Pri11] J Ronald Prins. Diginotar certificate authority breach “operation black tulip”. *Fox-IT*, November, page 18, 2011.
- [PSSW09] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 250–267. Springer, 2009.
- [PSZ18] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on OT extension. *ACM Trans. Priv. Secur.*, 21(2):7:1–7:35, 2018.
- [Qra20] Qrator Labs. This is how you deal with route leaks, 2 April 2020. <https://habr.com/en/company/qrator/blog/495260/> Accessed: 30 November 2020.

- [QUI] QUIC. <https://quicwg.org/> Accessed: 30 November 2020.
- [Rab05] Michael O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptol. ePrint Arch.*, 2005:187, 2005.
- [RABP17] Benjamin Rothenberger, Daniele Enrico Asoni, David Barrera, and Adrian Perrig. Internet kill switches demystified. In Cristiano Giuffrida and Angelos Stavrou, editors, *Proceedings of the 10th European Workshop on Systems Security, EUROSEC 2017, Belgrade, Serbia, April 23, 2017*, pages 5:1–5:6. ACM, 2017.
- [RBC⁺18] Andreas Reuter, Randy Bush, Ítalo S. Cunha, Ethan Katz-Bassett, Thomas C. Schmidt, and Matthias Wählisch. Towards a rigorous methodology for measuring adoption of RPKI route validation and filtering. *Comput. Commun. Rev.*, 48(1):19–27, 2018.
- [RIP11] RIPE NCC. RIPE NCC Blocks Registration in RIPE Registry Following Order from Dutch Police, 9 November 2011. <https://www.ripe.net/publications/news/about-ripe-ncc-and-ripe/ripe-ncc-blocks-registration-in-ripe-registry-following-order-from-dutch-police> Accessed: 30 November 2020.
- [RIP13] RIPE NCC. The RIPE NCC’s Case Against the State of the Netherlands Dismissed, 14 February 2013. <https://www.ripe.net/publications/news/about-ripe-ncc-and-ripe/ripe-nccs-case-against-the-state-of-the-netherlands-dismissed> Accessed: 30 November 2020.
- [RIP19] RIPE NCC. Closure of Members, Deregistration of Internet Resources and Legacy Internet Resources, 29 March 2019. <https://www.ripe.net/publications/docs/ripe-716> Accessed: 30 November 2020.
- [RIP20a] RIPE NCC. Accidental ROA Deletion, 2 April 2020. <https://www.ripe.net/support/service-announcements/accidental-roa-deletion> Accessed: 30 November 2020.
- [RIP20b] RIPE NCC. A First for the RIPE NCC: Seizure of the “Right to Registration of IPv4 Addresses” for the Recovery of Money, 2 October 2020. https://labs.ripe.net/Members/ciaran_byrne/seizure-of-the-right-to-registration-of-ipv4-addresses Accessed: 30 November 2020.

- [RN19] Stephen M. Ryan and Sam C. Neel. Taking a hard line on fraud, 13 May 2019. <https://teamarin.net/2019/05/13/taking-a-hard-line-on-fraud/> Accessed: 30 November 2020.
- [RW10] Raksha Reddy and Carl Wallace. Trust anchor management requirements. RFC 6024, RFC Editor, October 2010.
- [SA19] Nigel P. Smart and Younes Talibi Alaoui. Distributing any elliptic curve based protocol. In Martin Albrecht, editor, *Cryptography and Coding - 17th IMA International Conference, IMACC 2019, Oxford, UK, December 16-18, 2019, Proceedings*, volume 11929 of *Lecture Notes in Computer Science*, pages 342–366. Springer, 2019.
- [SAA⁺19] Muhammad Saad, Afsah Anwar, Ashar Ahmad, Hisham Alasmary, Murat Yuksel, and Aziz Mohaisen. Routechain: Towards blockchain-based secure and efficient BGP routing. In *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2019, Seoul, Korea (South), May 14-17, 2019*, pages 210–218. IEEE, 2019.
- [Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
- [Sho00] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 207–220. Springer, 2000.
- [SS01] Douglas R. Stinson and Reto Strobl. Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In Vijay Varadharajan and Yi Mu, editors, *Information Security and Privacy, 6th Australasian Conference, ACISP 2001, Sydney, Australia, July 11-13, 2001, Proceedings*, volume 2119 of *Lecture Notes in Computer Science*, pages 417–434. Springer, 2001.
- [SS10] Sooel Son and Vitaly Shmatikov. The hitchhiker’s guide to DNS cache poisoning. In Sushil Jajodia and Jianying Zhou, editors, *Security and Privacy in Communication Networks - 6th International ICST Conference, SecureComm 2010, Singapore, September 7-9, 2010. Proceedings*, volume 50 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 466–483. Springer, 2010.

- [SW17] Haya Shulman and Michael Waidner. One key to sign them all considered vulnerable: Evaluation of DNSSEC in the internet. In Aditya Akella and Jon Howell, editors, *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pages 131–144. USENIX Association, 2017.
- [SZ13] Thomas Schneider and Michael Zohner. GMW vs. yao? efficient secure two-party computation with low depth circuits. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, volume 7859 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2013.
- [Tal19] Talos Intelligence. DNS hijacking abuses trust in core internet service, April 17, 2019. <https://blog.talosintelligence.com/2019/04/seaturtle.html> Accessed: 30 November 2020.
- [Tin19] Mark Tinka. RPKI ROV & Dropping of Invalids – Africa, 09 April 2019. <https://www.mail-archive.com/apops@apops.net/msg00796.html> Accessed: 30 November 2020.
- [Unb19] Unbound Tech. blockchain-crypto-mpc, February 14, 2019. Version 1.0.2. <https://github.com/unbound-tech/blockchain-crypto-mpc>.
- [VDBvRDSP14] Gijs Van Den Broek, Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. DNSSEC meets real world: dealing with unreachability caused by fragmentation. *IEEE communications magazine*, 52(4):154–160, 2014.
- [vRJS16] Roland van Rijswijk-Deij, Mattijs Jonker, and Anna Sperotto. On the adoption of the elliptic curve digital signature algorithm (ECDSA) in DNSSEC. In *12th International Conference on Network and Service Management, CNSM 2016, Montreal, QC, Canada, October 31 - Nov. 4, 2016*, pages 258–262. IEEE, 2016.
- [WFGZ10] Haitao Wu, Zhenqian Feng, Chuanxiong Guo, and Yongguang Zhang. ICTCP: incast congestion control for TCP in data center networks. In Jaudelice Cavalcante de Oliveira, Maximilian Ott, Timothy G. Griffin, and Muriel Médard, editors, *Proceedings of the 2010 ACM Conference on Emerging Networking Experiments and Technology, CoNEXT 2010, Philadelphia, PA, USA, November 30 - December 03, 2010*, page 13. ACM, 2010.

- [WJLH06] David X. Wei, Cheng Jin, Steven H. Low, and Sanjay Hegde. FAST TCP: motivation, architecture, algorithms, performance. *IEEE/ACM Trans. Netw.*, 16(6):1246–1259, 2006.
- [WMK16] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. EMP-toolkit: Efficient MultiParty computation toolkit. <https://github.com/emp-toolkit>, 2016.
- [WRK17] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated garbling and efficient maliciously secure two-party computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 21–37. ACM, 2017.
- [WS19] P. Wouters and O. Sury. Algorithm implementation requirements and usage guidance for dnssec. RFC 8624, RFC Editor, June 2019.
- [Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986.
- [YW19] Christopher S Yoo and David A Wishnick. Lowering legal barriers to rpki adoption. *Faculty Scholarship at Penn Law*, 2035, 2019.
- [ZE15] Samee Zahur and David Evans. Obliv-c: A language for extensible data-oblivious computation. *IACR Cryptol. ePrint Arch.*, 2015:1153, 2015.
- [ZHS07a] Fangming Zhao, Yoshiaki Hori, and Kouichi Sakurai. Analysis of privacy disclosure in DNS query. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE 2007), 26-28 April 2007, Seoul, Korea*, pages 952–957. IEEE Computer Society, 2007.
- [ZHS07b] Fangming Zhao, Yoshiaki Hori, and Kouichi Sakurai. Two-servers PIR based DNS query scheme with privacy-preserving. In *The 2007 International Conference on Intelligent Pervasive Computing, IPC 2007, Jeju Island, Korea, 11-13 October 2007*, pages 299–302. IEEE Computer Society, 2007.
- [ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *Advances*

in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II, volume 9057 of *Lecture Notes in Computer Science*, pages 220–250. Springer, 2015.